Willem Jonker
Milan Petković (Eds.)

LNCS 6933

Secure Data Management

8th VLDB Workshop, SDM 2011
Seattle, WA, USA, September 2011
Proceedings

Springer

Lecture Notes in Computer Science 6933

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

Editorial Board

David Hutchison
Lancaster University, UK

Takeo Kanade
Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler
University of Surrey, Guildford, UK

Jon M. Kleinberg
Cornell University, Ithaca, NY, USA

Alfred Kobsa
University of California, Irvine, CA, USA

Friedemann Mattern
ETH Zurich, Switzerland

John C. Mitchell
Stanford University, CA, USA

Moni Naor
Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz
University of Bern, Switzerland

C. Pandu Rangan
Indian Institute of Technology, Madras, India

Bernhard Steffen
TU Dortmund University, Germany

Madhu Sudan
Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos
University of California, Los Angeles, CA, USA

Doug Tygar
University of California, Berkeley, CA, USA

Gerhard Weikum
Max Planck Institute for Informatics, Saarbruecken, Germany

Willem Jonker   Milan Petković (Eds.)

Secure Data Management

8th VLDB Workshop, SDM 2011
Seattle, WA, USA, September 2, 2011
Proceedings

Springer

Volume Editors

Willem Jonker
University of Twente
Faculty of Electrical Engineering,
Mathematics and Computer Science
P.O. Box 217, 7500 AE Enschede
The Netherlands
E-mail: willem.jonker@ictlabs.eu

Milan Petković
Philips Research Europe
High Tech Campus 34
5656 AE Eindhoven
The Netherlands
E-mail: milan.petkovic@philips.com

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-23555-9 e-ISBN 978-3-642-23556-6
DOI 10.1007/978-3-642-23556-6
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011934809

CR Subject Classification (1998): K.6.5, D.4.6, E.3, C.2, H.4, H.3

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

© Springer-Verlag Berlin Heidelberg 2011

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.
The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

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 year was the eighth edition of the VLDB Secure Data Management Workshop. The topic of data security remains an important area of research especially due to the steady growing proliferation of emerging data services such as cloud computing, location-based services, and health-related services. Confidentiality is the main driver for the research that covers topics such as privacy-enhancing technologies, access control, and search in encrypted data.

We received 19 submissions of which the Program Committee selected 10 papers to be presented at the workshop and included in the proceedings (approx. 50% acceptance rate). We hope the papers collected in this volume will stimulate your research in this area.

The regular papers in the proceedings are grouped into three sections. The first section focuses on privacy. The papers in this section present a study in privacy violations followed by application-driven work on algorithms. The second section focuses on data security in networks. The papers address issues related to management of confidential data that are stored in cloud and sensor networks. The third section groups a collection of more basic secure data management techniques that can find their way toward many different application settings.

We wish to thank all the authors of submitted papers for their high-quality submissions. We would also like to thank the Program Committee members as well as additional referees for doing an excellent review job. Finally, let us acknowledge Luan Ibraimi who helped in the technical preparation of the proceedings.

September 2011

Willem Jonker
Milan Petković

Organization

Workshop Organizers

Willem Jonker	EIT ICT Labs /University of Twente, The Netherlands
Milan Petković	Philips Research/Eindhoven University of Technology, The Netherlands

Program Committee

Imad Abbadi	Oxford University, UK
Gerrit Bleumer	Francotyp-Postalia, Germany
Ljiljana Brankovic	University of Newcastle, Australia
Sabrina De Capitani di Vimercati	University of Milan, Italy
Ernesto Damiani	University of Milan, Italy
Deng Mina	Philips Research, The Netherlands
Eric Diehl	Techicolor, France
Jeroen Doumen	Irdeto, The Netherlands
Csilla Farkas	University of South Carolina, USA
Eduardo Fernandez-Medina Paton	Universidad de Castilla-La Mancha, Spain
Elena Ferrari	Università degli Studi dell'Insubria, Italy
Simone Fischer-Hübner	Karlstad University, Sweden
Tyrone Grandison	IBM Research, USA
Dieter Gollmann	Technische Universität Hamburg-Harburg, Germany
Marit Hansen	Independent Centre for Privacy Protection, Germany
Min-Shiang Hwang	National Chung Hsing University, Taiwan
Mizuho Iwaihara	Waseda University, Japan
Sushil Jajodia	George Mason University, USA
Ton Kalker	Huawei, USA
Marc Langheinrich	University of Lugano (USI), Switzerland
Nguyen Manh Tho	Vienna University of Technology, Austria
Sharad Mehrotra	University of California at Irvine, USA
Stig Frode Mjølsnes	Norwegian University of Science and Technology, Norway
Eiji Okamoto	University of Tsukuba, Japan
Sylvia Osborn	University of Western Ontario, Canada

Günther Pernul	University of Regensburg, Germany
Birgit Pfitzmann	IBM Watson Research Lab, Switzerland
Bart Preneel	KU Leuven, Belgium
Kai Rannenberg	Goethe University Frankfurt, Germany
Ahmad-Reza Sadeghi	Darmstadt University, Germany
Andreas Schaad	SAP Labs, France
Jason Smith	Queensland University of Technology, Australia
Morton Swimmer	Forward-looking Threat Research, Trend Micro, Germany
Clark Thomborson	University of Auckland, New Zealand
Sheng Zhong	State University of New York at Buffalo, USA

Additional Referee

Carlos Blanco	Universidad de Castilla-La Mancha, Spain

Table of Contents

Quantifying Privacy Violations

Mishtu Banerjee, Rosa Karimi Adl, Leanne Wu, and Ken Barker

Advanced Database Systems Laboratory, Department of Computer Science,
University of Calgary, Calgary, Alberta, Canada
{smbanerj,rkarimia,lewu,kbarker}@ucalgary.ca

Abstract. Understanding privacy in a data storage environment has become of increasing interest to the data management and user communities over the past decade. Previous work has produced a number of definitions with greater or lesser specificity. The value of a particular definition can only be understood in light of how it helps us understand when a privacy violation occurs. This paper builds upon earlier work that defines privacy using a four-dimensional taxonomy with an inherent sense of increasing privacy exposure. This taxonomy is extended to formally capture the notions of (a) privacy violations, (b) the severity of a privacy violation, and (c) the likelihood of data providers ceasing to provide data due to privacy exposures. The privacy violation model developed here provides an operational framework to characterize and estimate privacy violation in a relational database system. It also allows one to calculate the consequences to the data provider of widening privacy policies. We describe a quantitative analysis of violations that captures discrepancies between the data collector's stated policies and practices in comparison to the data providers' data preferences. We demonstrate this analysis using a simple example and show how the accumulation of privacy violations can have a detrimental effect upon the data collector.

1 Introduction

Data systems of all varieties increasingly rely on the collection and storage of data from everyday users. As the volume and frequency of data collection and storage from individuals increases, privacy becomes an ever-present concern. The notion of privacy is a difficult one to articulate, and may cover a wide range of concerns when it comes to user data. For example, there exist privacy violations defined by legislation which are reliant on human factors, such as maintaining the ability of the data provider to access and update the information solicited from them; we draw the focus of the paper away from these types of violations for the present to concentrate on forms of data privacy which can be protected by using a data management system.

In a general sense, we find that the uses to which collected and stored data is put (the 'practices' of the data collector) often conflict with the preferences of the individuals who have provided such data, particularly with regards to the individual privacy of the data providers. This discrepancy has motivated a tremendous research effort in the field of data privacy.

© Springer-Verlag Berlin Heidelberg 2011

The majority of studies in the field of data privacy are concerned with a particular subclass of privacy violations, which may also be called ‘leakages’ or ‘inadvertent disclosures’, among several related terms. This subclass comprises a narrow set of privacy concerns, and is relatively straightforward to describe. The study of such violations is centered around the act of publishing a given set of data, and preventing an external party from being able to re-identify specific individuals whose data was included. Relatively few such studies also attempt to *define* the cause, nature and effect of such violations. We examine this question as a starting point to expand the range of violations with which our model can address, in the process applying a pre-existing privacy model to explore the concept of a privacy violation.

Legal and social definitions of privacy centre upon the maintenance of a transparent relationship between data collectors and data providers, not simply upon the risk of inadvertent disclosures when data is released or published. Privacy can be violated without data release or publication if data is used outside of the stated purpose for which it was collected, or retained beyond an agreed period. Privacy protection in this context requires the ability to verify, in a quantifiable manner, that information systems which house data from a provider conform to a stated privacy policy.

Furthermore, data providers vary in their privacy preferences. Thus, protection of individual privacy depends not only on a data collector conforming to its stated privacy policies, but also on how those privacy policies align with the preferences of individual providers. Given a clearly stated and enforced privacy policy, an individual who is very sensitive about their privacy may distrust the house more than an individual who is relatively unconcerned about their privacy. Privacy concerns of data providers are varied. They may include the access of data by unauthorized persons or for unauthorized purposes, use of data in more detail (such as including a person’s actual weight rather than a weight range), or the retention of data for an unspecified period in time. Such data provider concerns are common across a wide variety of applications, including healthcare, social networking, government records, or customer relationship management.

We build on these considerations of variability in data providers’ privacy concerns to make fundamental contributions to the issues of privacy in several ways. The key contributions of this paper are a model of privacy violations that provides definitions of: (1) privacy violations and a privacy–preserving database; (2) sensitivity to and severity of privacy violations; and (3) data provider default, where data providers cease to provide data due to privacy violations. These definitions can be put into practice in the context of relational database systems. Additionally, we (4) use the above definitions to examine the trade-offs between benefits to the house due to widening its privacy policy versus losses to the house via data providers defaulting from the system under a widened privacy policy.

1.1 Related Work

The best known approaches to prevent privacy violations have focussed on data release. k-Anonymity and its refinements [20] [14] [13] concern the release of tabular data in such a way that individuals are not identifiable. Differential privacy [3] [2] [4] attempts to guarantee that participation in a statistical database will not allow any third-party to learn any more about the individual (via that participation) than if the individual had not participated. A more recent approach [12] adapts methods from statistical decision theory to identify the privacy risk by incorporating both the probability of identifying an individual and the sensitivity of the disclosed information.

Two approaches are generally used to quantify data privacy. The first relies on social science to query data providers, potential or otherwise, on their beliefs with respect to certain privacy issues. One of the best known of these approaches compiles work performed by Westin [11]. However, such studies are specific to a certain population of data providers at a single point in time, and cannot be used to frame a general model. The other approach is based on information theory, in which the degree of information leakage from data released from the data store can be calculated. A typical example is provided by Ngoc *et al.* [15]. However, this approach is largely concentrated on specific problems, and cannot be used to build a general privacy model for any data repository.

Efforts have also been made to understand the protection of data privacy as an equilibrium between two different parties with conflicting interests. van Heerde *et al.* [7] describe how the process of setting a policy for data degradation can be described as an attempt to establish an equilibrium between the data provider and the data collector. Gianini and Damiani [6] and Ren and Xiao [17] both apply game theory to model privacy decisions with respect to external risks to privacy. The problems are described as zero-sum games between attackers and a data collector. While the former explores the problem in a location anonymization context, the latter focuses on finding an optimum level in a predefined generalization hierarchy. Preibusch [16] utilizes game theory concepts to demonstrate the interactions between data providers and a data collector in a dynamic privacy policy setting.

2 Modeling Privacy

'A Data Privacy Taxonomy' [1] introduces a model of privacy more comprehensive than had previously been offered in the literature. The taxonomy represents privacy as a point in a four-dimensional space, where each dimension represents a different privacy predicate: purpose, visibility, granularity and retention. Purpose refers to the uses for which data has been collected. Visibility describes the parties who will have access to the data while it is in storage. Granularity defines the specificity of data which will be revealed. Retention describes how long the data will be kept in storage.

Consider the act of collecting and storing private data as a form of gambling; this process incurs the risk of a privacy violation. The taxonomy therefore defines three kinds of 'players' in a data privacy scenario. There are *data providers* who provide raw data. There is the *house*, which is the organization that maintains a data repository. This is generally the party which stands to profit from any use of the data, regardless of whether the data provider's privacy has been protected. There are also *third-parties* who have access to the data maintained by the house. The interactions of these players constitute a privacy gamble, in which data providers exchange their data in return for services, and risk having their privacy violated by the house or third-parties.

The ideas in [1] have recently been applied to characterizing differences in privacy policy between different social networks [23]. An earlier work [22] demonstrated data providers are willing to provide more information to a house if they can provide it at coarser granularity rather than a specific atomic value. The corollary of this observation is that a data provider may provide less information if the house's policy requires it to be provided as precise atomic values. If the house's policy severely violates the data provider's privacy preferences, the data provider may even default; they will not participate, and contribute zero information to the system.

In the context of the taxonomy, other research efforts, particularly in the area of anonymization [20] [14] [13] and differential privacy [3] [2] [4] focus on granularity, and assume risk comes from forces external to the system. Our approach differs by focussing on privacy at the stages of data acquisition, storage and data access for a specified purpose. In the taxonomy, all four dimensions are called into play in defining privacy violations, the severity of privacy violations, and the likelihood data providers leave the system due to privacy violations.

Focus on external risks leads to a scenario where an honest house releases data in such a way that data provider privacy is protected. In this context, privacy protection is equated with maintaining the anonymity of individual records. Research with this focus provides an explicit attack model, defining the computational capabilities of third-parties given a particular anonymization protocol. The key issue in such scenarios is the risk of released data being used to identify individual data providers.

By contrast, focus on internal risks leads to the question of how a specific house policy (or the modification of an existing house policy) would affect a data provider, given their privacy preferences. A key issue in such scenarios is the degree to which house policies conform to the privacy concerns of data providers. When the privacy practices of the house do not align with the privacy preferences of the provider, the challenge becomes making the privacy practices of the house transparent enough that data providers can identify the areas where alignment has not been achieved, and quantify the resulting privacy violation. Automation of this procedure makes privacy violations auditable, so that data providers can continuously monitor the state of their privacy.

We focus on the first step towards data privacy transparency, by quantifying the degree to which house privacy policies conform to data provider privacy preferences. To do so, we need to bring together house privacy policies and data provider privacy preferences in such a way that it is possible to identify when they may be in conflict, in particular, those instances where a data provider's privacy may be violated. The privacy taxonomy [1] represents privacy as data points in a four-dimensional space. Implicit in this notion is the idea that a data provider's privacy preferences and a house's privacy policy can be represented as points in a geometric space. We make this idea explicit in such a way that we can formally define what it means for a data provider's privacy preferences to be violated by a house. We further investigate the consequences of such violations.

An explicit model of privacy violations following from the taxonomy has several benefits. First, such a model is one of the requirements in the construction of a privacy–preserving database. Secondly, it provides a tool for the analysis of privacy scenarios, to determine whether a given scenario may involve the risk of a privacy violation. Third, it is a first step to characterizing the dynamics of different parties (data provider, house, third-party) that may be in conflict with respect to privacy of data.

We now turn to the intuition behind our model, followed by our core assumptions. Next, we define features that distinguish the privacy perspective of the house from that of data providers. We then introduce formal definitions of the privacy violation of an individual data provider, from which we calculate the probability of a privacy violation. We develop a framework for characterizing the severity of privacy violations and use it to define the default of individual data providers, from which we calculate the probability of default. We illustrate this process with a simple example. Finally, we show the consequences of a house widening its privacy policy. Via doing so, the house increasingly violates the privacy preferences of data providers, which leads to a level of default of data providers that limits the gains to the house.

3 Privacy Violations

The privacy taxonomy provides a simple geometric view of a privacy violation. Figure 1 illustrates our intuition. We plot on the privacy taxonomy selected dimensions of a privacy policy for the house, and the corresponding components of the preferences for the data provider. Each privacy predicate contained by a house's privacy policy or a data provider's preference can be geometrically represented as a point in a privacy space. The points are denoted by privacy tuples. If, for any dimension defined by the taxonomy, the house's privacy policy does not form a box which can be completely bounded by those of the data provider's preferences, then we say a violation has occurred along that dimension. Part b) of Figure 1 shows a violation along a single dimension and part c) shows a violation along two dimensions.

We make the following assumptions:

1. The dimensions of privacy are orthogonal.
2. Values for the granularity, visibility and retention can be put into a total order, for the purposes of defining both the existence and severity of privacy violations.
3. Every datum is associated with a privacy tuple which is expressed as particular values for granularity, visibility, retention and purpose.
4. Purpose is treated differently from the other dimensions, and provides a grouping principle for individual privacy tuples. In that sense, purpose acts

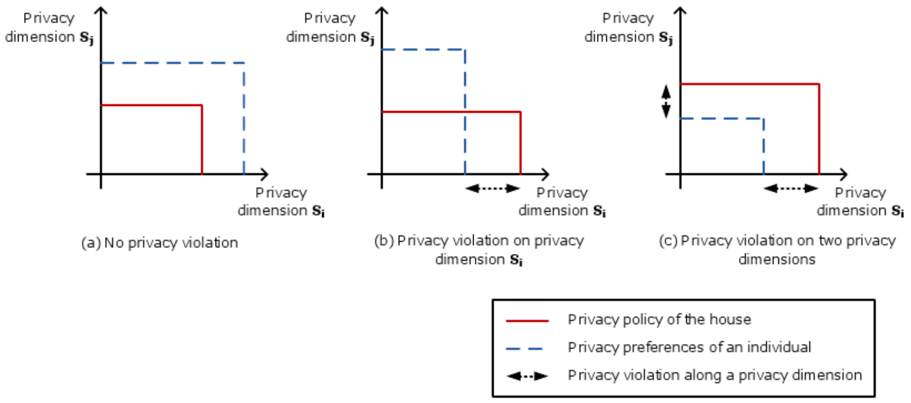

Privacy dimension S_j
Privacy dimension S_i
(a) No privacy violation
(b) Privacy violation on privacy dimension S_i
(c) Privacy violation on two privacy dimensions
Privacy policy of the house
Privacy preferences of an individual
Privacy violation along a privacy dimension

Fig. 1. Comparison between a single privacy preference tuple and a privacy policy tuple over two privacy dimensions S_i and S_j. Privacy preference and privacy policy tuples are defined for the same data attribute and share the same purpose.

like a categorical variable, and our key assumption here is simply that different purposes are distinguishable. There is ongoing research which intends to give the concept of purpose a set of structured semantics by arranging purpose into a lattice [5] or as a series of interrelated actions[9]. If this approach leads to a total ordering on the purpose dimension, then in this case we could treat purpose as any other privacy dimension without changing our approach.

5. For simplicity, every tuple in a data table is considered to represent a single data provider. The model below can be extended to deal with situations where multiple records may exist in the same table for a given data provider, by specifying the appropriate one-to-many relationships between a data provider and tuples in a table. Such extensions will depend on the specifics of the entities being modelled.

4 Defining the House and Data Providers

To model the relationship between data and privacy we need to bring together the data in a database, and privacy associated with that data. We distinguish the privacy policies held by a house from the privacy preferences held by an individual data provider. The essential idea we are building up to is that a privacy violation occurs when the privacy policy of a house exceeds the privacy preferences for a data provider around a specific datum the data provider supplies.

Figure 2 visually represents the notation we introduce below. We define the policies of the house, the preferences of a data provider, the database and the specific data to which privacy policies and preferences may apply. Superscripts are used for data attributes. Subscripts are used for individual data providers. Square brackets are used for dimensions of privacy.

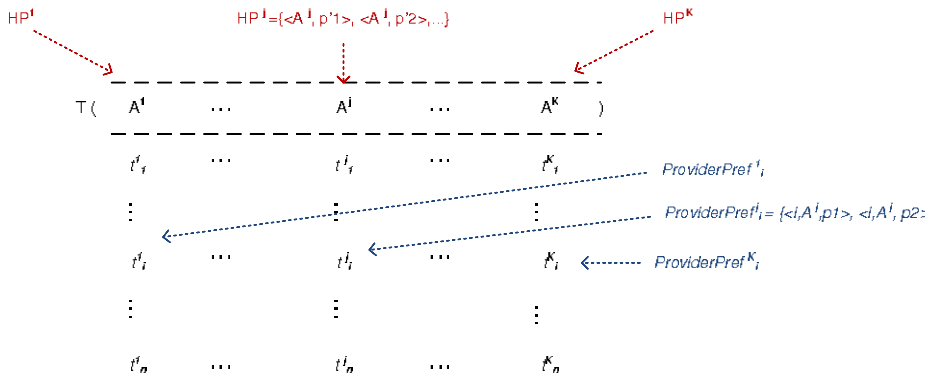


Fig. 2. Visual Representation of Privacy Violation Model Notation

Let the data table of private information be: $T = \{t_1, ..., t_n\}$ where n is the number of records in the table.

The relation schema can be defined as:

$T(A^1 \in D^1, A^2 \in D^2, ..., A^K \in D^K)$ where $A^j \in A$ is the j^{th} attribute with domain D^j.

We denote the data from the ith data provider for the jth attribute as t_i^j.

Next let us define the privacy policies (house) and the privacy preferences (data provider). There are $dim = 4$ privacy dimensions: visibility (V), purpose (Pr), granularity (G) and retention (R). The set of all privacy tuples P is the cross product of the privacy dimensions.

$$P = Pr \times V \times G \times R. \tag{1}$$

The house may have multiple privacy tuples associated with the jth attribute in its database. The ith data provider has a set of privacy preference tuples associated with each datum t_i^j. When we want to refer to a particular element of a privacy tuple, p, we use the notation $p[dim]$, so that $p[dim]$ is the actual value for dimension dim of a privacy tuple.

A privacy policy (or preference) is a collection of privacy tuples associated with each attribute (or datum).

Let $Policy$ be the set of all possible policy tuples for a house.

$$Policy = \{< a, p > | a \in A \land p \in P\}. \tag{2}$$

A particular house policy HP is defined as a subset of $Policy$:

$$HP \subseteq Policy. \tag{3}$$

From HP we can extract the house's privacy policy for collecting attribute A^j as follows:

$$HP^j = \{< a, p > \mid < a, p > \in HP \land a = A^j\}. \tag{4}$$

Every potential data provider has a set of privacy preferences for every piece of data he/she provides. We define the privacy preferences of an individual i by $ProviderPref_i$ as follows:

$$ProviderPref_i = \{\langle i, a, p\rangle | i \in n \wedge a \in A \wedge p \in P\}. \tag{5}$$

Similar to the house's privacy policy, we define privacy preferences of individual i on data value t_i^j as follows:

$$ProviderPref_i^j = \{\langle i, a, p\rangle | \langle i, a, p\rangle \in ProviderPref_i \wedge a = A^j\}. \tag{6}$$

5 Definition of Privacy Violation

A privacy violation can now be defined in terms of the above notation.

Definition 1. *Let w_i be a variable denoting whether privacy of data provider i has been violated or not. The value of w_i can be determined as follows:*

$$\begin{aligned} w_i = 1 \Leftrightarrow\ & \exists\ \langle i, a, p\rangle \in ProviderPref_i, \\ & \exists \langle a, p'\rangle \in HP, \\ & \exists\ dim \neq Pr\ \ s.t. \\ & p[Pr] = p'[Pr]\ \wedge \\ & p[dim] < p'[dim] \\ w_i = 0\quad\ & otherwise. \end{aligned} \tag{7}$$

Notice that if there exists a purpose $pr = p'[Pr]$, such that individual i has not explicitly specified any preference for, we assume that the individual does not prefer to reveal her information for purpose pr and we add the tuple $\langle i, a, \langle pr, 0, 0, 0\rangle\rangle$ to $ProviderPref_i$. As a result, in the process of assessing a privacy violation, we compare privacy policy and privacy preferences of each purpose separately. Privacy violations occur when, within a given purpose, the house policy tuple for a datum exceeds the data provider's privacy preference on one of the other three dimensions (Visibility, Granularity, Retention).

Given a formal definition of a privacy violation, it is straightforward to define the probability of a privacy violation in a database. We take a relative frequency approach to the definition of probabilities ([19]:pp. 1-3, [18]:pp. 30-33.) where for a repeated series of trials in which τ is the number of trials, and $\tau(A)$ is the number of trials in which the event A occurred, the fraction of trials in which A occurred, $\frac{\tau(A)}{\tau}$, tends towards the probability of A, $P(A)$. So $P(A) \sim \frac{\tau(A)}{\tau}$ for a large series of trials.

Definition 2. *Let each trial be the random selection of a data provider from the database and the determination of whether their privacy has been violated. Let $\tau(W)$ be the number of trials in which privacy is violated for a randomly selected data provider. Let τ be the total number of trials. Let $P(W)$ be the probability that a randomly selected data provider's privacy will be violated. There are N data providers.*

$$P(W) \sim \frac{\tau(W)}{\tau} \sim \frac{\Sigma_i w_i}{N}. \tag{8}$$

With this definition, we can simply define a privacy–preserving database (PPDB) in terms of $P(W)$.

Definition 3. *Let an α-PPDB be a database where $P(W)$ is not above a threshold α.*

$$P(W) \leq \alpha. \tag{9}$$

6 Severity of Privacy Violation

The definition of privacy violation provided in Section 5, explains a violation as a binary variable which only denotes whether a violation has occurred or not. The definition is therefore incapable of measuring severity of a violation. We believe that severity of a violation depends on both sensitivity (of datum, attribute, and privacy dimensions) and also the amount of violation along each axis.

6.1 Sensitivity

Use of private information in a manner that extends beyond an individual's privacy preferences causes varying amounts of discomfort to the data provider. The amount of discomfort depends upon the type of information and the value the provider associates with a data item. The sensitivity factors in our privacy violation measurement incorporate these differences in evaluating the severity of a privacy violation. The relative sensitivities of data attributes and data values are highly tied to social norms and need to be defined according to a careful survey of the population of providers.

For example, Westin [21] recognizes financial and health information as the most sensitive data attributes. Kobsa [10] also claims that personal preferences, demographics, life style information are less sensitive compared to financial, purchase related, online behavior, religion, political party identification, and occupation information. For each of these information types, as the data value negatively deviates from the social norms the sensitivity of the value increases [8]. As the range of data solicited from everyday users broadens and its uses become even more myriad, further study of how data providers are sensitive to privacy violations involving such data becomes a pressing need.

Sensitivity factors for each purpose in a private database can be defined as:

$$Sensitivity = \langle \sigma,\ \Sigma \rangle. \tag{10}$$

Where $\sigma = \langle \sigma_1,\ \sigma_2,\ ..., \sigma_n \rangle$ is the vector of sensitivity factors for each data provider and $\Sigma = \langle \Sigma^1,\ \Sigma^2,\ ...,\ \Sigma^k \rangle$ denotes the vector of sensitivity values (defined as an integer number) associated with each data attribute.

Every data provider, i, has a sensitivity factor σ_i which denotes the sensitivity he/she associates with each data value and each dimension of privacy. In other words, $\sigma_i = \langle \sigma_i^1,\ \sigma_i^2,\ , ...,\ \sigma_i^k \rangle$, where σ_i^j explains the sensitivity element that data provider, i, associates with datum provided for attribute A^j. These sensitivity elements can be described as:

$$\sigma_i^j = \langle s_i^j,\ s_i^j[V],\ s_i^j[G],\ s_i^j[R] \rangle. \tag{11}$$

Where s_i^j is the sensitivity of the data value t_i^j, and the rest of the elements reflect the sensitivity of a violation along visibility, granularity, and retention dimensions.

6.2 Privacy Violation Measurement

To measure the severity of a violation along each dimension, we assume that the visibility, granularity, and retention values are specified via a numerical value in preference and privacy tuples. Since we have assumed total ordering for the values along these dimensions, numerical values can simply be chosen to reflect the orderings.

To begin with, we define the function $diff : N \times N \rightarrow Z$ to measure the difference between a preference tuple and a privacy tuple along visibility, granularity, or retention dimension:

$$diff(p,\ P) = \begin{cases} P - p & if\ P > p \\ 0 & otherwise \end{cases} \tag{12}$$

Now we can assess the amount of conflict between a privacy preference tuple, $pref = \langle i, a, p \rangle$, and a privacy policy tuple, $Pol = \langle a', p' \rangle$. To do so, we first define the notion of comparability; a preference tuple and a privacy tuple are comparable if they are both associated with the same attribute and have the same purpose. Therefore we have:

$$comp(pref,\ Pol) = \begin{cases} 0 & if\ a \neq a' \\ 0 & if\ p[Pr] \neq p'[Pr] \\ 1 & otherwise \end{cases} \tag{13}$$

Given a pair of privacy preference tuples, $pref = \langle i, a, p \rangle$, and privacy policy tuple, $Pol = \langle a', p' \rangle$, the privacy conflict can be measured as:

$$conf(pref,\ Pol) = comp(pref,\ Pol) \times \sum_{dim \in \{V,G,R\}} diff(p[dim],\ p'[dim]) \times \Sigma^{a'} \times s_i^a \times s_i^a[dim]. \tag{14}$$

Therefore, if the two tuples are comparable, the violation along each dimension is evaluated (using $diff$ function) and weighted by the sensitivity of the attribute ($\Sigma^{a'}$), then sensitivity of the data value (s_i^a), and the sensitivity of the dimension ($s_i^a[dim]$). All of these sensitivities are tied to a specific purpose.

To evaluate the amount of privacy violation for each data provider, i, we need to compare all of his/her privacy preference tuples against all privacy policy tuples in the house. The total privacy violation for data provider, i, is the summation of all mutual conflicts between these two sets:

$$Violation_i = \sum_{pref \in ProviderPref_i} \sum_{Pol \in HP} conf(pref,\ Pol). \tag{15}$$

Finally, the amount of privacy violations conducted by the house is defined as the summation of violations over all data providers:

$$Violations = \sum_{i=1}^{n} Violation_i. \tag{16}$$

7 Definition of Data Provider Default

Let us consider the circumstances under which a privacy violation may lead to a data provider defaulting, *i.e.* leaving the system and no longer contributing data to the house.

Breadth. A data provider may default if there are privacy violations on many attributes of their information.

Depth. A data provider may default if there is a particularly large violation on a single attribute or datum (particularly if the attribute is considered sensitive or the datum is sensitive for that data provider).

The formulation of $Violation_i$ combines both aspects of privacy violation. Given sufficient privacy violations, a data provider i will default, and no longer contribute data to the house.

Definition 4. *For a data provider i, let v_i be a threshold value for $Violation_i$ above which data provider i will default from the database. Let $default_i$ represent whether an individual i will default.*

$$default_i = \begin{cases} 1 & if\ Violation_i > v_i \\ 0 & otherwise \end{cases} \tag{17}$$

Given a formal definition of default, it is straightforward to define the probability of data provider default in a database.

Definition 5. *Let each trial be the random selection of a data provider from the database and the determination of whether that data provider will default from the database. Let $\tau(Default)$ be the number of trials in which a randomly selected data provider defaults from the database. Let τ be the total number of trials. Let $P(Default)$ be the probability that a randomly selected data provider will default due to the severity of privacy violations.*

$$P(Default) \sim \frac{\tau(Default)}{\tau} \sim \frac{\Sigma_i default_i}{N}. \tag{18}$$

In the next section we provide a simple example of the interplay between house gains due to expanding its privacy policies versus losses due to data providers defaulting from the system.

8 A Simple Example

To illustrate how our proposed method measures privacy violations, we provide a simple example of a private data collection situation. In our example the private data table only has two attributes: $A = \{Age, Weight\}$. For simplicity we assume a single privacy tuple associated with each attribute and a single privacy preference tuple for each data item. We also assume that data providers' privacy preferences are defined in such a way that the house's privacy tuple on Age does not violate anyone's preferences. As a result we only focus on privacy violations on attribute $Weight$.

House policy on attribute $Weight$ is defined as: $HP^{Weight} = \{\langle Weight, pr, v, g, r\rangle\}$ where pr, v, g, and r are some specific values along purpose, visibility, granularity, and retention dimensions. Moreover, assume that we have $\Sigma^{Weight} = 4$ as the sensitivity of attribute $Weight$. Let `Alice`, `Ted`, and `Bob` be the only data providers in the system. Privacy preferences, sensitivities, and violation threshold of each data provider are described in Table([1]).

Table 1. Privacy Preferences of Data providers on attribute $Weight$

Data provider	$ProviderPref_i^{Weight}$	σ_i^{Weight}	v_i	w_i
`Alice`	$\{< Weight, pr, v+2, g+1, r+3 >\}$	$\langle 1,1,2,1\rangle$	$v_{Alice} = 10$	$w_{Alice} = 0$
`Ted`	$\{< Weight, pr, v+2, g-1, r+2 >\}$	$\langle 3,1,5,2\rangle$	$v_{Ted} = 50$	$w_{Ted} = 1$
`Bob`	$\{< Weight, pr, v, g-1, r-1 >\}$	$\langle 4,1,3,2\rangle$	$v_{Bob} = 100$	$w_{Bob} = 1$

As can be seen in the table, privacy of `Ted` is violated on attribute $Weight$ along granularity dimension and privacy of `Bob` is violated on attribute $Weight$ along both granularity and retention dimensions. Since all the privacy preferences are comparable with the privacy policy, we calculate the conflicts as follows:

$$conf(pref,\ Pol) = \sum_{dim \in \{V,G,R\}} diff(p[dim],\ p'[dim]) \times \Sigma^{a'} \times s_i^a \times s_i^a[dim]. \quad (19)$$

$$\begin{aligned} &conf(ProviderPref_{Alice}^{Weight},\ \langle Weight, pr, v, g, r\rangle) = 0 \\ &conf(ProviderPref_{Ted}^{Weight},\ \langle Weight, pr, v, g, r\rangle) = 1*4*3*5 = 60 \\ &conf(ProviderPref_{Bob}^{Weight},\ \langle Weight, pr, v, g, r\rangle) = 1*4*4*3 + 1*4*4*2 = 80 \end{aligned} \quad (20)$$

Since each data provider has only a single privacy preference tuple for $Weight$ data and the house has a single privacy tuple attached to this attribute, the conflicts are also the amount of violations. The defaulting behaviour of the data providers (based on their violation threshold) are as follows:

$$Violation_{Alice} = 0 < 10 \Rightarrow default_{Alice} = 0 \quad (21)$$

$$Violation_{Ted} = 60 > 50 \Rightarrow default_{Ted} = 1 \tag{22}$$

$$Violation_{Bob} = 80 < 100 \Rightarrow default_{Bob} = 0 \tag{23}$$

Notice that although `Bob's` privacy is violated along two dimensions (compared to `Ted`), the combination of his sensitivities and violation threshold are set such that he stays in the system. However, since `Ted` believes that a violation along granularity dimension is very sensitive and has lower violation threshold, he will default and leave the system. Consequently, $P(Default)$ is:

$$P(Default) = \frac{0+1+0}{3} = \frac{1}{3} \tag{24}$$

9 Expansion of the Privacy Policies for a House

In a commercial setting, information provided to the house by data providers often defines a revenue stream in terms of its value to third-parties. It is in the house's interest to expand its privacy policies, so it has more information to sell per data provider. However, that trend is counter-balanced by the increased likelihood of data providers defaulting as the house expands its privacy policy.

In a non-commercial setting, utility is provided by other means than revenue. Utility could be in terms of cost savings, or it could be in terms of social goods, such as public safety, public security or public health. However utility is valued, we can formulate the additional utility a house must acquire per user via an expansion of its privacy policy to justify any defaults due to that change in policy.

Let $N_{current}$ be the number of data providers currently in the system.

Let U be the utility per data provider i currently obtained by the house. Then,

$$Utility_{current} = N_{current} \times U. \tag{25}$$

Let us assume that currently, no data providers have defaulted; i.e. all $Violation_i$ are less than the critical v_i. The house can justify an expansion of its privacy policy only if it provides additional utility. Let N_{future} be the number of data providers after the house has expanded its privacy policy.

$$N_{future} = N_{current} - \Sigma_i default_{i_{future}}. \tag{26}$$

Let T be the additional utility above U per data provider available to the house due to the expansion of its privacy policy.

$$Utility_{future} = N_{future} \times (U + T). \tag{27}$$

The condition to justify a house in expanding its privacy policy is:

$$Utility_{future} > Utility_{current}. \tag{28}$$

$$N_{future} \times (U + T) > N_{current} \times U. \tag{29}$$

Gather revenues per data provider on the LHS and population sizes on the RHS.

$$\frac{U+T}{U} > \frac{N_{current}}{N_{future}}. \tag{30}$$

If we solve for T:

$$T > U(\frac{N_{current}}{N_{future}} - 1). \tag{31}$$

We have now defined the minimum amount of extra utility per data provider that must accrue to the house to compensate for any losses due to defaults after expanding its privacy policy. In a commercial setting, the utilities could be in terms of revenue to the house. In a non-commercial settings, the utilities could represent cost savings due to an expansion of house policy. However, in other situations, it may be inappropriate to value utility in monetary terms either as revenue gains or cost savings. In those cases, other measures of utility would have to be devised.

This example is very simple with some strong assumptions to simplify our analysis. We are assuming that utilities are expressed simply per data provider, and not dependent on the specific population sizes, or 'influences' amongst data providers. We are assuming that these utilities can be strictly valued. We are assuming that data providers have free choice as to whether they will participate in a database. We assume that expansions of house privacy policies are not ameliorated by the provision of incentives. Given such assumptions, the basic nature of the trade-offs between increased utility and data provider default are illustrated. Effectively, the increases of a house's privacy policies can be seen as negative utilities with respect to data providers which eventually lead data providers to default. The house is strictly limited in how much it can expand its privacy policies and economically benefit. Weakening of these assumptions leads naturally to a game theoretic setting where one can examine the balance between the competing interests of a house and its data providers.

10 Contributions of the Framework and Next Steps

Providing data establishes a relationship between a set of data providers and a house. Data providers are inclined to stay in that relationship to the extent they trust the house to protect their privacy. So, the degree to which data provider's privacy preferences are protected by house privacy policies must be quantifiable as a first step. Further steps towards trust would include verification via an audit framework to ensure that the house is adhering to its stated privacy policies.

Our key goal in developing this privacy violations model is to understand how to evaluate whether house privacy policies conform to data provider privacy preferences. Understanding the conditions under which privacy violations, and data provider defaults occur allows us to reason about the dynamics of changing privacy policies in databases. This can apply to issues that often frustrate privacy-conscious users such as frequently changing privacy policies on social networking sites. While privacy preservation has been defined in specific contexts including public statistical databases [3] and modified values in a table before release [20], our privacy violations model provides the first demonstration of the capabilities required to facilitate a privacy-preserving database applicable to a broad range of relational databases.

The violations model described here can be used to calculate the probability of a violation. It is also possible to develop *'what if'* scenarios that modify a house's privacy policies with respect to data provider default. Thus, if a particular default level is explicitly adopted, the database can be demonstrably shown to be an $\alpha - PPDB$. The model also contributes by identifying how house policies and data provider privacy preferences must be brought into alignment to determine (a) whether violations have occurred, (b) the degree of violation, and (c) the impact on the house, based on the number of users defaulting due to violations.

Incorporating the privacy violation model and approach developed here into legacy systems is also possible. Future work will explore how this might be accomplished pragmatically. For example, in the absence of explicit tracking of providers' privacy preferences or knowledge of the specific values v_i at which data providers default, the model identifies the quantities that require estimation. Long-term observation of a particular house and its population of users, or survey questions (see [22]) can be used to identify the number of users who will default as a house expands its privacy policy. This in turn can be used to empirically construct a cumulative distribution function of the number of defaults as the house expands its privacy policies. This function can then be used to examine particular house scenarios projected by the modification of its privacy policies.

A general privacy violation model provides rich opportunities for future work. The legacy system issue mentioned earlier and a detailed understanding of how to model various scenarios using techniques such as game theory suggest themselves immediately. These questions are already being considered but the opportunities are much broader. The current thrust should immediately consider the issues associated with developing an initial prototype of the $\alpha - PPDB$ based on the model described here. This will undoubtedly lead to several open problems and test the feasibility of the violations model expeditiously. The model can then be refined with an eye to capturing the challenging problem of real-time dynamics occurring between a house and a set of (possibly very heterogeneous) data providers or end users.

Significant work remains not just in extending the limits of this model, but in strengthening the model via evaluation and further study. A first step would be in producing a simulation using a sample dataset to show that our model has the properties claimed by this paper; later steps would see implementation of algorithms which could use the model to check for privacy violations in "real-life" scenarios. Further work (in the realm of the social sciences) also remains to determine the feasibility of identifying values for sensitivity for each data provider, and whether these values can be accurately assessed.

Finally, though certainly not exhaustively, this work has only considered a traditional relational database model. Extending it to other popular structures such XML will undoubtedly identify new opportunities and challenges. This may involve changing the violation model itself. It would also reveal new ways of approaching the kind of data that should be collected to help ensure that data providers are protected by only collecting data at the appropriate level of granularity which will afford much better protection for end users. Nonetheless, this violations model provides a framework for anwering many important questions which have not yet been posed in the literature.

References

1. Barker, K., Askari, M., Banerjee, M., Ghazinour, K., Mackas, B., Majedi, M., Pun, S., Williams, A.: A data privacy taxonomy. In: Sexton, A.P. (ed.) BNCOD 26. LNCS, vol. 5588, pp. 42–54. Springer, Heidelberg (2009)
2. Dwork, C.: Ask a better question, get a better answer a new approach to private data analysis. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 18–27. Springer, Heidelberg (2006)
3. 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)
4. Dwork, C.: Differential privacy: A survey of results. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 1–19. Springer, Heidelberg (2008)
5. Ghazinour, K., Barker, K.: Capturing p3p semantics using an enforceable lattice-based structure. In: Proceedings of the 4th International Workshop on Privacy and Anonymity in the Information Society, PAIS 2011, pp. 4:1–4:6. ACM, New York (2011), http://doi.acm.org/10.1145/1971690.1971694
6. Gianini, G., Damiani, E.: A game-theoretical approach to data-privacy protection from context-based inference attacks: A location-privacy protection case study. In: Jonker, W., Petkovic, M. (eds.) SDM 2008. LNCS, vol. 5159, pp. 133–150. Springer, Heidelberg (2008)
7. van Heerde, H., Fokkinga, M., Anciaux, N.: A framework to balance privacy and data usability using data degradation. In: International Conference on Computational Science and Engineering, CSE 2009, vol. 3, pp. 146–153 (29-31, 2009)
8. Huberman, B.A., Adar, E., Fine, L.R.: Valuating privacy. IEEE Security & Privacy 3, 22–25 (2005)
9. Jafari, M., Fong, P.W., Safavi-Naini, R., Barker, K., Sheppard, N.P.: Towards defining semantic foundations for purpose-based privacy policies. In: Proceedings of the first ACM conference on Data and application security and privacy, CODASPY 2011, pp. 213–224. ACM, New York (2011), http://doi.acm.org/10.1145/1943513.1943541
10. Kobsa, A.: Privacy-enhanced web personalization. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 628–670. Springer, Heidelberg (2007)
11. Kumaraguru, P., Cranor, L.F.: Privacy indexes: A survey of Westin's studies. Tech. rep., Carnegie Mellon University (2005)
12. Lebanon, G., Scannapieco, M., Fouad, M.R., Bertino, E.: Beyond k-anonymity: A decision theoretic framework for assessing privacy risk. Trans. Data Privacy 2, 153–183 (2009), http://portal.acm.org/citation.cfm?id=1744063.1744064
13. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: IEEE 23rd International Conference on Data Engineering, ICDE 2007, pp. 106–115 (2007)

14. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L-diversity: Privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data 1 (March 2007), http://doi.acm.org/10.1145/1217299.1217302
15. Ngoc, T.H., Echizen, I., Komei, K., Yoshiura, H.: New approach to quantification of privacy on social network sites. In: International Conference on Advanced Information Networking and Applications (2005)
16. Preibusch, S.: Implementing privacy negotiations in e-commerce (2005)
17. Ren, Y., Xiao, Z.: A privacy data release method based on game theory. In: 2nd International Conference on e-Business and Information System Security (EBISS) 2010, pp. 1–4 (May 2010)
18. Renỳi, A.: Probability Theory. Dover Press, New York (2007)
19. Rozanov, Y.: Probability Theory: A Concise Course. Dover Press, New York (1977)
20. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems 10(5)
21. Westin, A.F.: Social and political dimensions of privacy. Journal of Social Issues 59(2), 431–453 (2003)
22. Williams, A., Barker, K.: Controlling inference: avoiding p-level reduction during analysis. In: Proceedings of the fifth Australasian symposium on ACSW frontiers, ACSW 2007, vol. 68, pp. 193–200. Australian Computer Society, Inc., Darlinghurst (2007), http://portal.acm.org/citation.cfm?id=1274531.1274554
23. Wu, L., Majedi, M., Ghazinour, K., Barker, K.: Analysis of social networking privacy policies. In: EDBT 2010: Proceedings of the 2010 EDBT/ICDT Workshops, pp. 1–5. ACM, New York (2010)

On-the-Fly Generalization Hierarchies for Numerical Attributes Revisited

Alina Campan, Nicholas Cooper, and Traian Marius Truta

Department of Computer Science, Northern Kentucky University,
Highland Heights, KY 41099, USA
{campana1,trutat1}@nku.edu coopern1@mymail.nku.edu

Abstract. Generalization hierarchies are frequently used in computer science, statistics, biology, bioinformatics, and other areas when less specific values are needed for data analysis. Generalization is also one of the most used disclosure control technique for anonymizing data. For numerical attributes, generalization is performed either by using existing predefined generalization hierarchies or a hierarchy-free model. Because hierarchy-free generalization is not suitable for anonymization in all possible scenarios, generalization hierarchies are of particular interest for data anonymization. Traditionally, these hierarchies were created by the data owner with help from the domain experts. But while it is feasible to construct a hierarchy of small size, the effort increases for hierarchies that have many levels. Therefore, new approaches of creating these numerical hierarchies involve their automatic/on-the-fly generation. In this paper we extend an existing method for creating on-the-fly generalization hierarchies, we present several existing information loss measures used to assess the quality of anonymized data, and we run a series of experiments that show that our new method improves over existing methods to automatically generate on-the-fly numerical generalization hierarchies.

Keywords: anonymization, k-anonymity, hierarchies for quasi-identifier numerical attributes.

1 Introduction and Motivation

Generalization hierarchies are frequently used in computer science, statistics, biology, bioinformatics, and other areas when less specific values than the original ones are needed for data analysis. The term generalization hierarchy is used in data privacy and anonymity community and more recently in data mining community. Generalization hierarchies are commonly called taxonomies (biology, bioinformatics, statistics, etc.) or concept hierarchies (data mining and data warehousing).

These hierarchies provide the foundation of roll-up and roll-down operations in a data warehousing system [13]. In data mining, generalization hierarchies are used in various data mining techniques such as characteristic rule mining [9] classification [19], association rule mining [11], and clustering [5, 7]. Other areas of computer science such as machine learning [20], data integration [34], object-oriented databases [12], and intrusion detection [18] also use generalization hierarchies. Recently,

© Springer-Verlag Berlin Heidelberg 2011

generalization hierarchies received a renewed attention in the data privacy field. Statistical disclosure control community used global/local recoding (a close substitute of a generalization hierarchy) as a disclosure control technique for protecting datasets against de-identification [41]. In the data anonymity community, the seminal papers of Sweeney [35] and Samarati [33] reinforced the use of generalization as a powerful and useful technique to achieve *k*-anonymity [33, 35].

Generalization consists in replacing the actual value of an attribute with a less specific, more general value that is faithful to the original [36]. In general, generalization is based on a *domain generalization hierarchy* (*DGH*) associated to that attribute. Such a generalization hierarchy is usually provided by a domain expert based on the attribute characteristics. A second hierarchy, called *value generalization hierarchy* (*VGH*), represents all values from different domains/levels of the domain generalization hierarchy and their ancestor/descendant relationships. Fig. 1 shows two examples of DGHs and VGHs for attributes *country* and *gender*.

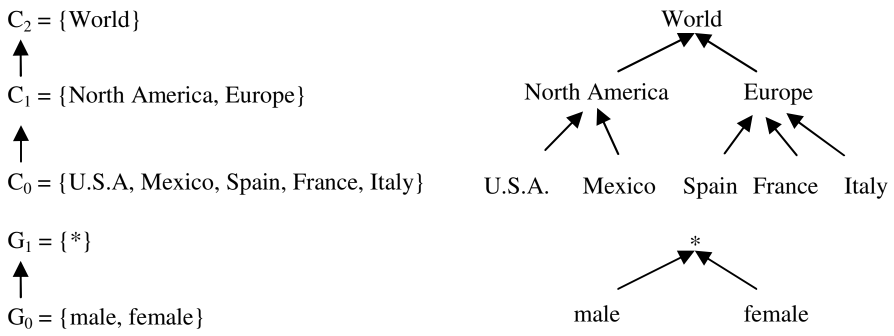


Fig. 1. DGHs and VGHs for attributes *country* and *gender*

Generalization is one of the most used disclosure control technique for anonymizing data. It is applied to microdata sets in order to avoid de-identification of individuals. *Microdata* represents a series of tuple, each tuple containing information on an individual unit such as a person or organization [41]. We call the original microdata initial microdata (*IM*). Due to existing regulations in various areas (such as Health Insurance Portability and Accountability Act, HIPAA [14]), *IM* should be released for use by a third party only after the owner of the data has masked it to limit the possibility of disclosure. We call the final microdata ready for release, the masked microdata (*MM*).

Generalization was initially used for *categorical* attributes with *predefined* DGHs and VGHs constructed by the domain experts [36]. Generalization was next extended for *numerical* attributes either by using *predefined hierarchies* [16] or a *hierarchy-free model* [23]. While generalization of numerical attributes using predefined hierarchies is similar to the generalization for categorical attributes, the generalization of numerical attributes without generalization hierarchies is based on determining generalization intervals/bins during the anonymization process based on an optimization criterion (such as minimizing information loss). Based on how generalization intervals are created, that hierarchy-free generalization for numerical attributes helps minimizing the information loss that occurs in the masking process, and might perform in that respect better than using a predefined hierarchy. Still, there are situations when hierarchy-free generalization is not suitable for anonymization. First, creating generalization intervals during the anonymization process does not guarantee that those intervals are disjoint (in many situations these intervals will overlap) and this will create difficulties in analyzing the resulting masked microdata. For example, the values 12, 17, 23 for the attribute *age* can be generalized to the interval [12 – 23], and the values 16, 34 for the same attribute can be generalized to [16 – 34]. The reason why the grouping is not based on the order of values (the first group in that case would be 12, 16, 17; the second group 23, 34) is because there are other attributes involved in the anonymization process and the values of those attributes will impact the creation of groups; due to their influence, the overlapping generalization intervals for the numerical attribute are preferred for a smaller overall information loss in the anonymized data. If these overlaps are not desired in the resulting masked microdata, due to the nature of the application, then the data owner should use hierarchies during the generalization process. Second, certain data anonymity models, such as constrained *k*-anonymity (which relies on boundaries imposed on the amount of generalization allowed in the anonymization process) [30] and personalized anonymity (which uses guarding nodes as boundaries for the sensitive information) [39] require pre-existing hierarchies for numerical attributes.

Based on the above considerations, we conclude that there are situations when using hierarchies for numerical attributes during the anonymization process cannot be avoided. Traditionally, the generalization hierarchies were created by the data owner with help from the domain experts. But while it is feasible to construct a hierarchy of small size, the effort increases for large hierarchies. The manual construction of a generalization hierarchy might cause problems such as erroneous classifications or omissions of concepts [17]. Usually, creating and understanding a hierarchy for categorical attributes is easier than for numerical attributes: the values of the categorical attribute are well established, discrete, have a natural hierarchical organization, while numerical attributes have many values and not very often have a natural hierarchical structure. Moreover, a domain expert will not be able to capture the data characteristics when designing a generalization hierarchy, and this will likely lead to creating masked microdata where the information loss is high. Using a "good" hierarchy in the anonymization process significantly impacts the quality of the anonymized microdata; depending on how well the hierarchy fits the distribution and grouping of the attribute's values in the microdata set. Using on-the-fly hierarchies created based on data characteristics will help in creating a better-quality masked microdata.

Automatic generation methods for creating generalization hierarchies in data mining community (usually called concept hierarchies) exist for both categorical and numerical values. There are only a few studies for categorical attributes since, as mentioned before, these hierarchies are in general easier to create by human experts [22]. For numerical hierarchies, many techniques to generate automatic hierarchies are proposed in the literature. The binning method, which partitions numbers in equal ranges or equal frequencies, is reviewed in [13]. Extensions to this method include histogram analysis and numeric clustering [10, 13]. Other approaches that try to locate better cutting points are based on recursive binary discretization [4], minimum description length [6], entropy-based discretization [32], chi-square test [21, 26], relaxation error [5], and attribute-oriented induction [15]. All these methods focus on preprocessing data before applying data mining techniques, and they are not tailored to data anonymity. Still, we selected two such approaches introduced by Han and Fu [10] and by Chu and Chiang [5] for our experimental comparison.

A method to generate on-the-fly numerical hierarchies for anonymizing data is introduced in [3]. A hierarchical clustering agglomerative approach [37] is used to construct a hierarchy based on the distance between already created nodes in the generalization hierarchy of the target attribute.

Our research contributions in this paper are as follows.

First, we improve the existing method for creating on-the-fly hierarchies for numerical attributes introduced in [3]. Our new method will replace the agglomerative selection approach based on minimal distance between nodes with the selection of two neighbor nodes that, combined, will create the smallest possible node (in a sense that we will describe later) at that step. This improved method is presented in Section 2.

Second, we discuss how the generated hierarchies are used during anonymization and we present several existing information loss measures that assess the information lost in the generalization of numerical quasi-identifier attribute values.

Third, we perform a series of experiments on the Adult dataset [17]. We generate *k*-anonymous masked microdata sets using the on-the-fly generalization hierarchies created based on our new method, using the existing method presented in [3], using a set of predefined hierarchies, and without using hierarchies (hierarchy-free generalization). We also create anonymized datasets using hierarchies generated with two existing methods used for dynamic generation of numerical hierarchies in data mining [5, 10]. The quality of the resulting datasets is compared with respect to the information loss measures discussed in Section 3. These information loss measures' values are dependent on the hierarchies used to perform generalization and on the anonymization algorithm used. To compare the quality of generalization hierarchies, we use the same anonymization algorithm (introduced in [2]) for all our generated datasets.

The paper ends with conclusions and suggestions for future work.

2 On-the-Fly Hierarchies for Numerical Attributes

The initial microdata ($\mathcal{IM}$) is described by a set of attributes that are classified into three categories: *identifier* attributes such as *Name* and *SSN* that can be used to identify a tuple; *quasi-identifier* attributes such as *ZipCode* and *Sex* that may be known by an intruder; and *confidential* or *sensitive* attributes such as *Diagnosis* and *Income* that are assumed to be unknown to an intruder.

In the released dataset (called *masked microdata* and labeled $\mathcal{MM}$) only the quasi-identifier and confidential attributes are preserved; identifier attributes are removed as a prime measure for ensuring data privacy. Although direct identifiers are removed, an intruder may use record linkage techniques between externally available datasets and the quasi-identifier attributes values from the masked microdata to glean the identity of individuals. To avoid this possibility of disclosure, one frequently used solution is to further process (modify) the initial microdata through generalization and suppression [36] of quasi-identifier attributes values, so that to enforce the *k*-anonymity property for the masked microdata. In order to rigorously and succinctly express *k*-anonymity property, we use the following concept:

Definition 1. (*QI-Cluster*): Given a microdata, a ***QI-cluster*** consists of all the tuples with identical combination of quasi-identifier attribute values in that microdata.

We define *k*-anonymity based on the minimum size of all *QI*-clusters.

Definition 2. (*K-Anonymity Property*): The ***k-anonymity property*** for a $\mathcal{MM}$ is satisfied if every *QI*-cluster from $\mathcal{MM}$ contains *k* or more tuples.

Unfortunately, *k*-anonymity protects only against identity disclosure and it fails to protect confidential information against attribute disclosure [29, 38]. As a result, several anonymity models were introduced to increase the protection of confidential information of individuals in the released datasets. Some of the most known extensions of *k*-anonymity include *l*-diversity [29], *p*-sensitive *k*-anonymity [38], (α, *k*)-anonymity [42], *t*-closeness [25], (ε, *m*)-anonymity [24], l^+-diversity [27], and (τ, λ)-uniqueness [40].

Generalization is one of the most used techniques to create a masked microdata that satisfies not only *k*-anonymity but also any of the improved anonymization models. For a fair comparison of the quality of generated masked microdata sets with various generalization hierarchies, the same anonymization model must be used. In this paper we decided to use *k*-anonymity for our comparison. While a different anonymization model may increase the information loss (due to a stronger privacy requirement, the utility is expected to drop), we expect that the information loss for various generalization hierarchies will keep for other models the relative proportion they have for *k*-anonymity.

Let *K* be the numerical quasi-identifier attribute for which we construct a generalization hierarchy. We denote by $V = \{v_1, v_2, \ldots, v_m\}$ the distinct values of *K* in the dataset $\mathcal{IM}$. Each one of these values can have one or more occurrences in $\mathcal{IM}$. If more than one numerical quasi-identifier attribute needs on-the-fly hierarchies, they are constructed individually, one attribute at a time.

The method to create on-the-fly hierarchies is described next. The construction of the hierarchy starts with a set of *m* nodes, one node for each of the *m* unique values of the attribute *K*. These nodes will become the leaves of the domain value hierarchy labeled $\mathcal{H}_K$ for the attribute *K*. Next, the hierarchy is built from leaves to root, by merging at each step two nodes that will create the smallest possible node at that step. The generalization hierarchy is completely built when all values are combined into a single node, the root of the hierarchy. The resulting hierarchy is a tree, called a dendrogram [13], which is usually not balanced, and which can have its leaves on any level under the root.

We will define next the size of a node and how two nodes are merged in our approach.

Definition 3. *(a node in the numerical hierarchy).* Each node in $\mathcal{H}_K$, leaf or internal, is characterized by two values: the *minimum* (*min*) and *maximum* (*max*) numerical values represented by the node.

For a leaf node created for the value *v*, *min* and *max* are the same value (*v*). A node will be represented as $X = [min, max]$. We will denote by *v* both a value of *K* and its associated leaf node.

Definition 4. *(size of a node).* We compute the size of a node $X = [min, max]$ as $size(X) = max – min$.

Definition 5. *(adjacent nodes).* During the construction of a hierarchy, two nodes $X_i = [min^i, max^i]$ and $X_j = [min^j, max^j]$ are called adjacent if they do not have yet any ancestors (in other words these nodes were not yet used in merging) and there is no value from *K* between the two nodes (in other words the interval $(min(max^i, max^j), max(min^i, min^j))$ does not contain any value from *K*).

Definition 6. *(merge two nodes).* Two adjacent nodes $X_i = [min^i, max^i]$ and $X_j = [min^j, max^j]$ are merged into a new node $Y = merge(X_i, X_j) = [min(min^i, min^j), max(max^i, max^j)]$.Both X_i and X_j are made descendants of *Y* when merged. The nodes X_i and X_j are selected such that the resulting node (*Y*) will have the smallest possible size at that time.

We give next the pseudocode for the generalization algorithm for constructing a numerical attribute's hierarchy.

```
Algorithm Improved On-The-Fly Hierarchy (IOTF) is
 Input: $\mathcal{IM}$, attribute $K$
 Output: $\mathcal{H}_K$
 Extract from $\mathcal{IM}$ the leaf nodes in $\mathcal{H}_K$,
 $V = \{v_1, v_2, …, v_m\}$;
 each $v_i \in V$ has $v_i.min = v_i.max =$ value $v_i$;
 $\mathcal{H}_K = V$;
 Repeat
  Find $X_i, X_j \in V$ such that
   $X_i, X_j$ are adjacent and // see Definition 5
   $\forall X, Y \in V, size(merge(X_i, X_j)) \le size(merge(X, Y))$
   // In other words, $size(merge(X_i, X_j))$ is minimized
   // Merge two adjacent nodes that create the smallest new node
   $X_{new} = merge(X_i, X_j)$;
   Make $X_{new}$ parent in $\mathcal{H}_K$ for $X_i$ and $X_j$;
   $V = V - \{X_i, X_j\} \cup \{X_{new}\}$;
 Until ($|V| = 1$);
 The remaining node in $V$ is the root of $\mathcal{H}_K$;
End On-The-Fly Hierarchy.
```

In the above algorithm, the size of the current set of nodes, *V*, is reduced by one when two nodes are merged, and after *m*-1 iterations, only one node will remain in the set. This node becomes the root of the hierarchy. The hierarchies produced by this algorithm are shaped as binary trees and can be very deep, due to how they are created – they can actually have a maximum of m-1 levels. In is worth noting that at every iteration, the nodes from the current set of nodes are completely disjoint. In the generated hierarchy any initial value has a unique path from its corresponding leaf to the root. This prevents one problem that exists with hierarchy-free generalization (described in Section 1). Examples of hierarchies constructed with this algorithm are presented in Section 4.

The complexity of the *NumericalHierarchy* algorithm is $O(m^2)$. This is because, in each merging step, the two nodes to be merged can only be adjacent nodes in the list of current nodes V. The nodes in V are kept sorted based on their *max* value (any value from the node can be used in this ordering since the nodes are disjoint). Consequently, finding the pair of nodes in V that when merged create the smallest node implies comparing $|V|$-1 pairs of nodes. As the size of V decreases from m to 1, the overall cost is $O(\sum_{l=1}^{m-1} l) = O(m^2)$.

3 Information Loss Measures Used in Data Anonymity

To measure the quality of masked microdata we use and adapt several well known information loss (*IL*) / data utility measures. Since our on-the-fly generalization is applicable to numerical attributes only, we present in this section these information loss measures with the assumption that all quasi-identifier attributes are numerical. We exclusively limit quasi-identifiers to homogeneous combinations of numerical attributes, with or without hierarchies, to isolate and study the impact on masked microdata quality of using different types of numerical hierarchies in the anonymization process.

We use the following notations in this section:

- $QI = \{K_1, K_2,\ldots, K_p\}$ – the set of p numerical quasi-identifiers for the initial microdata, $\mathcal{IM}$.
- s – the number of quasi-identifier attributes for which we use hierarchy-free generalization. We agree that these attributes are the first s in the set QI ($\{K_1, K_2,\ldots, K_s\}$). Consequently, the set $\{K_{s+1}, K_{s+2},\ldots, K_p\}$ represent the quasi-identifier attributes that are generalized using hierarchies. Note that when $s = 0$, all quasi-identifier attributes have hierarchies and when $s = p$ all attributes are generalized using hierarchy-free generalization.
- n – the number of tuples from $\mathcal{IM}$.
- $cl = \{t_1, t_2,\ldots, t_q\}$ – a set of q tuples from $\mathcal{IM}$.
- $\mathcal{S} = \{cl_1, cl_2, \ldots, cl_u\}$ – a complete and disjoint partition of $\mathcal{IM}$ (every tuple from $\mathcal{IM}$ belongs to exactly one cluster from the partition).
- $t_r\,|QI = (t_r^1, t_r^2, \ldots, t_r^p)$, for all $r = 1..q$; $t_r\,|QI$ denotes the relational projection operation of a tuple t_r on the set of attributes QI.
- $[min^k(cl), max^k(cl)] = [min(t_1^k, t_2^k,\ldots, t_q^k), max(t_1^k, t_2^k,\ldots, t_q^k)]$ for all $k = 1..p$. This interval represents the generalization interval of the cluster cl for the attribute K_k when hierarchy-free generalization is used.
- $\mathcal{H}_{Kk}$ – the generalization hierarchy of the attribute K_k.
- $root(\mathcal{H}_{Kk})$ – the root node of $\mathcal{H}_{Kk}$.
- $anc^k(cl)$ – the generalization node in $\mathcal{H}_{Kk}$ for the cluster cl. This node is the first

common ancestor for all values form the cluster *cl* with respect to the attribute K_k. This node represents the interval [$anc^k(cl).min$, $anc^k(cl).max$] (see Definition 3). We also use $size(anc^k(cl)) = anc^k(cl).max - anc^k(cl).min$ as per Definition 4.

To achieve *k*-anonymity, $\mathcal{IM}$ is partitioned into clusters of size at least *k*. Each such cluster is generalized to the corresponding *QI*-cluster using either hierarchy-free generalization or hierarchy (predefined or on-the-fly)-based generalization for each quasi-identifier attribute. This process will lead to loss of information in $\mathcal{MM}$ compared to $\mathcal{IM}$.

The first information loss measure we present in Definitions 7 and 8 was previously presented in [3] and it extends the measure previously introduced in [2] by assessing the information loss in hierarchies where leaf nodes are situated at different levels.

Definition 7. *(cluster information loss due to generalization).* The information loss caused by generalizing a cluster *cl* to the same "tuple" (these tuples form a *QI*-cluster in $\mathcal{MM}$), denoted by *IL*(*cl*), is defined as follows:

$$IL(cl) = |cl| \times \left[\sum_{k=1}^{s} \frac{max^k(cl) - min^k(cl)}{max^k(IM) - min^k(IM)} + \sum_{k=s+1}^{p} \frac{size(anc^k(cl))}{size(root(H_{Kk}))}\right]$$

Definition 8. *(normalized total information loss).* The ***normalized total information loss*** for a partition into clusters, $\mathcal{S}$, of the initial microdata set, $\mathcal{IM}$, is the sum of the information loss for all clusters in $\mathcal{S}$ divided to the number of tuples from $\mathcal{IM}$ times the number of quasi-identifier attributes. Formally:

$$NTIL(IM, S) = \frac{\sum_{j=1}^{u} IL(cl_j)}{n \cdot p},$$

The maximum value for *NTIL* is 1, and it corresponds to the case when all tuples in $\mathcal{IM}$ would have each quasi-identifier attribute generalized to the interval that covers all of its values in the set, or, respectively, generalized to the root value of its value generalization hierarchy. The minimum value (0) is obtained when $\mathcal{MM}$ is the same as $\mathcal{IM}$ (there was no generalization performed).

The next two information loss measures presented in Definitions 9 and 10 are based on Minkowski-norms on group extents and they are introduced in [8].

Definition 9. *(normalized information loss – average-extent metric).* The ***normalized information loss based on average extent metric*** for a partition into clusters, $\mathcal{S}$, of the initial microdata set, $\mathcal{IM}$, is defined as follows:

$$NIL_1(IM, S) = \left[\sum_{j=1}^{u}\left(\sum_{k=1}^{s} \frac{max^k(cl_j) - min^k(cl_j)}{max^k(IM) - min^k(IM)} + \sum_{k=s+1}^{p} \frac{size\left(anc^k(cl_j)\right)}{size\left(root(H_{Kk})\right)}\right)\right] \Big/ p \cdot u$$

NIL_1 is similar to *NTIL* except it does not take into account the size of clusters from the partition $\mathcal{S}$. The range of values for NIL_1 is [0, 1], and the boundaries are also met for no generalization (*NIL*$_1$ = 0) and generalization to the root (*NIL*$_1$ = 1), respectively.

Definition 10. *(normalized information loss – maximum-extent metric).* The ***normalized information loss based on maximum extent metric*** for a partition into clusters, $\mathcal{S}$, of the initial microdata set, $\mathcal{IM}$, is defined as follows:

$$NIL_{\infty}(IM,S) = \left[\sum_{k=1}^{s} \max_{j=1,u}\left(\frac{max^k(cl_j) - min^k(cl_j)}{max^k(IM) - min^k(IM)}\right) + \sum_{k=s+1}^{p} \max_{j=1,u}\left(\frac{size\left(anc^k(cl_j)\right)}{size\left(root(H_{Kk})\right)}\right)\right] \Big/ p$$

NIL$_\infty$ is considering the maximum information loss per attribute between all clusters which is averaged for all quasi-identifier attributes and normalized to [0, 1]. While the value 0 is also obtained when there is no generalization, the value 1 can be obtained more easily, for instance it is enough if only a cluster is generalized to the root (or maximum interval, for hierarchy-free generalization) for all attributes. This value can also be obtained if for any quasi-identifier attribute, there is a cluster that generalizes that attribute to the root.

The last two information loss measures we present in Definitions 11 and 12 are based on *discernability metric* (*DM*) [1] and *average cluster size metric* (*AVG*) [23]. These measures are not normalized to [0, 1].

Definition 11. *(discernability metric).* The ***discernability metric (DM)*** assigns to each tuple from $\mathcal{IM}$ a penalty that is determined by the size of the cluster containing that tuple:

$$DM(IM,S) = \sum_{j=1}^{u}\left(|cl_j|\right)^2$$

Definition 12. *(average cluster size metric).* The ***average cluster size metric (AVG)*** is defined as follows:

$$NAVG(IM,S) = \frac{n}{u \cdot k}$$

4 Experimental Results

For our experiments, we selected the anonymization algorithm called *greedy k-member clustering* presented in [2]. This algorithm works by creating clusters of tuples from $\mathcal{IM}$, of size *k* or more. These clusters will be then generalized to the same tuple, forming a *QI*-cluster in the $\mathcal{MM}$. The clusters are created one at a time, starting from a seed tuple and absorbing one new tuple at a time, until the cluster has *k* tuples. The new tuple selection criterion is based on an objective function. The objective function in our case is the *NTIL* function; therefore, a new tuple is added to a cluster labeled *cl* if it produces a local minimum increase of *IL*(*cl*) (see Definition 7).

To assess the performance of the new proposed on-the-fly generalization method of hierarchies for numerical attributes, we used the Adult dataset [31]. This dataset is the de-facto benchmark for many data anonymization problems and it consists of 45,422 tuples. Since we want to compare the generalization hierarchies for numerical attributes we will restrict our experiments to the following 3 numerical quasi-identifier attributes: *age*, *education_num*, *hours_per_week*. As we already mentioned before, we considered all of the quasi-identifiers to be numerical, as to avoid the categorical ones to impact in any way the anonymization process and the quality of the masked microdata.

We performed experiments with six settings for the above mentioned quasi-identifier attribute set:

- Each attribute had a generalization hierarchy dynamically created with the method introduced in this paper. We refer to it as *IOTF* (improved on-the-fly) method.
- Each attribute had a generalization hierarchy dynamically created with the related method introduced in [3]. We call this method *OTF* (on-the-fly) method.
- Each attribute had *predefined* hierarchies. These hierarchies are the same as in [3].
- Each attributes did not have hierarchies (i.e. hierarchy-free generalization).
- Each attribute had a generalization hierarchy dynamically created with a method used in data mining for concept hierarchies introduced in [10]. In the algorithm to generate hierarchies from [10] we use a threshold value of 4 and a fan-out value of 5. We call this method *Han* based on the first author name.
- Each attribute had a generalization hierarchy dynamically created with a method used in data mining for concept hierarchies introduced in [5]. In the algorithm to generate hierarchies from [5] we use a threshold value of 2. We use the first author's name, *Chu*, to refer to this method.

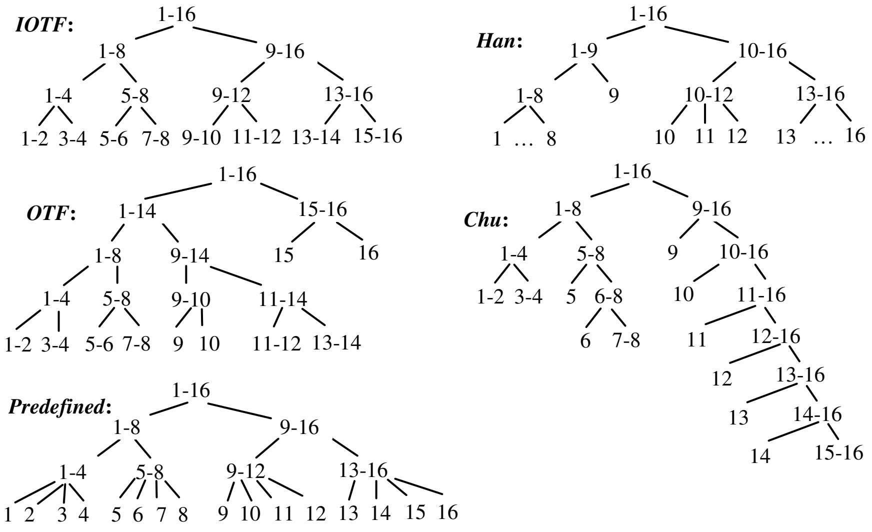


Fig. 2. VGHs for attribute *education_num* generated using *IOTF*, *OTF*, *Predefined*, *Han*, and *Chu* methods

We present in Fig. 2 the generated value generalization hierarchies for the attribute *education_num* (we selected this attribute as it has the smallest number of distinct values; similar hierarchies were generated for *age* and *hours_per_week* attributes using all five methods). In Fig. 2, due to space limitation some of the single value leafs are not shown.

In each setting, we anonymized the microdata set using the same algorithm ([2]), for all possible k values in the range 2 - 20. It is worth noting that in all experiments we either use hierarchies ($s = 0$) or a hierarchy-free approach ($s = p$) for all three quasi-identifier attributes (see Section 3 for definitions of s and p). For each experiment we computed all measures presented in Section 3: *NTIL*, NIL_1, NIL_∞, *DM*, and *AVG*.

Fig. 3 presents comparatively the normalized total information loss (*NTIL*) and normalized information loss based on average-extent metric (NIL_1) for all six cases, for the even values of k we considered in our experiments ($k = 2, .., 20$). It can be seen that the *IOTF* method of generating on-the-fly hierarchies outperform the other four methods based on generated or predefined hierarchies (*OTF*, *Predefined*, *Han*, and *Chu*) and as expected it does not perform as well as hierarchy-free generalization. However, as presented in Section 1, hierarchy-free generalization is not applicable in all anonymization scenarios. Out of the five generated or predefined hierarchy methods, *Han* and *Predefined* perform the worse because they do not use binary hierarchies, and therefore the generalization will create larger intervals faster than in the other methods. *Chu* and *OTF* methods produce results that are close to *IOTF*, however *IOTF* performed better with respect to *NTIL* and NIL_1 in all scenarios. The reason why *Chu* method performs reasonably well is because it uses a top down-approach in which intervals are split based on a measure (called relaxation error) that considers the value frequencies and the distance between values [5].

Fig. 4 presents comparatively the discernability metric (*DM*) and average cluster size (*AVG*) for all six cases, for even values of k considered in our experiments ($k = 2, .., 20$). The results are similar with the ones for *NTIL* and NIL_1 measures. Han and Predefined methods perform worse than the other methods, and as expected, hierarchy-free generalization performs the best. However in this case, there is almost a tie between the other three methods. For discernability metric values, out of 18 experiments ($k = 2, 3, \ldots, 20$), *IOTF* outperformed *OTF* and *Chu* 7 times, while *OTF* and *Chu* had the best result 6 times each. For average cluster size metric, *IOTF* had the best result 9 times, *OTF* 5 times, and *Chu* also 5 times. The reason why the proposed algorithm is not a clear winner for these two measures is because they do not consider the size of the created clusters, but only their number. As described in Section 2, the *IOTF* algorithm minimizes the size of newly created intervals, and this will contribute to smaller size clusters but not necessarily to fewer clusters.

We did not include a depiction with the results for NIL_∞ because, when using hierarchies, in almost all cases one cluster was generalized to the entire range for each attribute, and therefore the NIL_∞ measure is almost all the time 1. The only three cases (out of 95, five hierarchy-based methods and 19 values of k) when NIL_∞ was not equal to 1 are: ($k = 5$, *IOTF*), ($k = 6$, *IOTF*), and ($k = 3$, *Chu*). The reason why this measure is almost all the time 1 is the chosen anonymization algorithm. *K-member-clustering* [2] is a greedy algorithm that at the end will create large clusters (large not as number of members, but large with respect to our definition of size), and since NIL_∞ considers

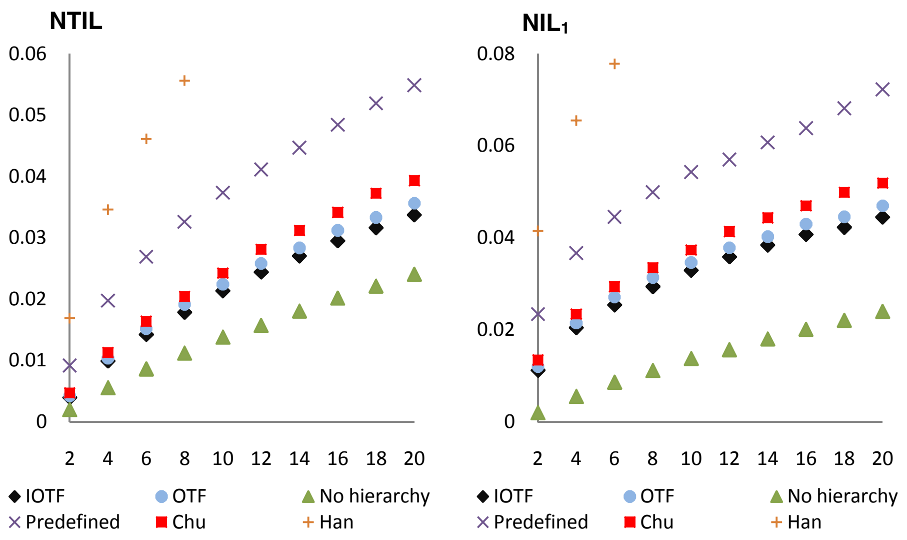


Fig. 3. *NTIL* and *NIL*$_1$ for k = 2, 4, .., 20 (even values) using five types of generalization hierarchies (*IOTF*, *OTF*, *Predefined*, *Chu*, and *Han*) and hierarchy-free generalization

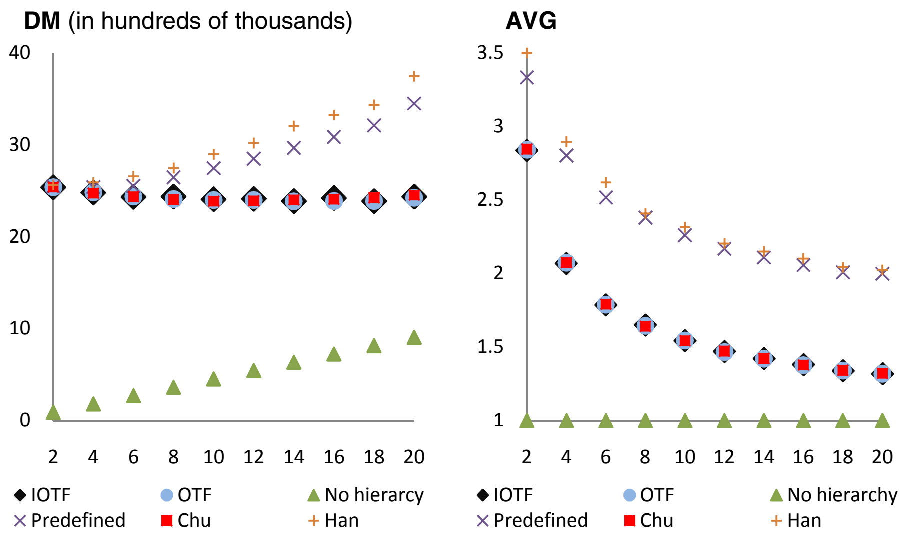


Fig. 4. *DM* and *AVG* for k = 2, 4, .., 20 (even values) using five types of generalization hierarchies (*IOTF*, *OTF*, *Predefined*, *Chu*, and *Han*) and hierarchy-free generalization

the maximum size intervals between those clusters, the results will always be 1 or close to 1. For the same reason, NIL_∞ measure was close to 1 (but not equal) for all 19 cases of hierarchy-based generalization.

5 Conclusions and Future Work

We introduced in this paper a new method for dynamically creating hierarchies for numerical quasi-identifier attributes. The resulting hierarchies represent a valid alternative to predefined hierarchies, and their usage generally results in good quality masked microdata, with reasonable information loss. Our new method clearly outperforms existing approaches to generate on-the-fly numerical hierarchies with respect to two information loss measures, normalized total information loss (*NTIL*) and normalized information loss based on average-extent metric (NIL_1). The proposed method had similar or slightly better results for the other three information loss measures, namely normalized information loss based on maximum-extent metric (NIL_∞), discernability metric (*DM*), and average cluster size (*AVG*), when compared with two other methods to create on-the fly hierarchies (*OTF* and *Chu*). On-the-fly hierarchies can be easily produced when hierarchies are necessary, instead of forcing the user to artificially develop ones that might not reflect the properties of the data, therefore negatively impacting the quality of the masked microdata.

As future work, we intend to investigate other anonymization algorithms that generate k-anonymous or l-diverse [29] masked microdata with respect to how well they perform using on-the-fly generalization hierarchies generated with the proposed method.

References

1. Bayardo, R.J., Agrawal, R.: Data Privacy through Optimal k-Anonymization. In: Proceedings of the IEEE International Conference of Data Engineering (ICDE), pp. 217–228 (2005)
2. Byun, J.W., Kamra, A., Bertino, E., Li, N.: Efficient k-Anonymity using Clustering Techniques. CERIAS Technical Report 2006-10 (2006)
3. Campan, A., Cooper, N.: On-the-Fly Hierarchies for Numerical Attributes in Data Anonymization. In: Jonker, W., Petković, M. (eds.) SDM 2010. LNCS, vol. 6358, pp. 13–25. Springer, Heidelberg (2010)
4. Catlett, J.: On Changing Continuous Attributes into Ordered Discrete Attributes. In: Kodratoff, Y. (ed.) EWSL 1991. LNCS, vol. 482, pp. 164–177. Springer, Heidelberg (1991)
5. Chu, W.W., Chiang, K.: Abstraction of High Level Concepts from Numerical Values in Databases. In: Proceedings of the Knowledge Discovery in Data Mining Workshop (KDD 1994), pp. 133–144 (1994)
6. Fayyad, U.M., Irani, K.B.: Multi-interval Discretization of Continuous-valued Attributes for Classification Learning. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 1022–1027 (1993)
7. Fischer, D.: Improving Inference through Conceptual Clustering. In: Proceedings of the National Conference on Artificial Intelligence (AAAI 1987), vol. 2, pp. 461–465 (1987)
8. Ghinita, G., Karras, P., Kalinis, K.: A Framework for Efficient Data Anonymization under Privacy and Accuracy Constraints. ACM transactions on database Systems 34(2) (2009)
9. Han, J., Cai, Y., Cercone, N.: Data-Driven Discovery of Quantitative Rules in Relational Databases. IEEE Transactions on Knowledge and Data Engineering 5(1), 29–40 (1993)

10. Han, J., Fu, Y.: Dynamic Generation and Refinement of Concept Hierarchies for Knowledge Discovery in Databases. In: Proceedings of the Knowledge Discovery in Data Mining Workshop (KDD 1994), pp. 157–168 (1994)
11. Han, J., Fu, Y.: Discovery of Multiple-level Association Rules from Large Databases. In: Proceedings of the Very Large Database Conference (VLDB 1995), pp. 420–431 (1995)
12. Han, J., Nishio, S., Kawano, H., Wang, W.: Generalization-based Data Mining in Object-oriented Databases using an Object Cube Model. Data and Knowledge Engineering 25, 55–97 (1998)
13. Han, J., Kamber, M.: Data Mining - Concepts and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2006)
14. HIPAA: Health Insurance Portability and Accountability Act (2002), `http://www.hhs.gov/ocr/hipaa`
15. Hsu, C.: Extending Attribute-oriented Induction Algorithm for Major Values and Numeric Values. Expert Systems with Applications 27, 187–202 (2004)
16. Iyengar, V.: Transforming Data to Satisfy Privacy Constraints. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 279–288 (2002)
17. Jing, Y., Croft, W.B.: An Association Thesaurus for Information Retrieval. Technical Report #94-17. University of Massachusetts at Amherst, Amherst, MA (1994)
18. Julisch, K.: Clustering Intrusion Detection Alarms to Support Root Cause Analysis. ACM Transactions on Information and System Security (TISSEC) 6(4), 443–471 (2003)
19. Kamber, M., Winstone, L., Gong, W., Cheng, S., Han, J.: Generalization and Decision Tree Induction: Efficient Classification in Data Mining. In: Proceedings of the International Workshop on Research Issues on Data Engineering (RIDE 1997), pp. 111–120 (1997)
20. Kaufman, K.A., Michalski, R.S.: A Method for Reasoning with Structured and Continuous Attributes in the INLEN-2 Multistrategy Knowledge Discovery System. In: Proceedings of the Knowledge Discovery in Data Mining Conference (KDD 1996), pp. 232–237 (1996)
21. Kerber, R.: Chimerge: Discretization of Numeric Attributes. In: Proceedings of the International Conference on Artificial Intelligence (AAAI 1992). MIT Press, Cambridge (1992)
22. Lee, S., Huh, S.Y., McNiel, R.D.: Automatic Generation of Concept Hierarchies using Wordnet. Expert Systems with Applications Journal 35(3), 1132–1144 (2008)
23. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Mondrian Multidimensional *K*-Anonymity. In: Proceedings of the IEEE International Conference of Data Engineering, Atlanta, Georgia (2006)
24. Li, J., Tao, Y., Xiao, X.: Preservation of Proximity Privacy in Publishing Numerical Sensitive Data. In: Proceedings of the ACM SIGMOD, pp. 473–486 (2008)
25. Li, N., Li, T., Venkatasubramanian, S.: *T*-Closeness: Privacy Beyond *k*-Anonymity and *l*-Diversity. In: Proceedings of the 23rd International Conference on Data Engineering (IEEE ICDE 2007), pp. 106–115 (2007)
26. Liu, H., Setiono, R.: Feature Selection via Discretization. IEEE Transactions on Knowledge and Data Engineering 9(4), 642–645 (1997)
27. Liu, J.Q., Wang, K.: On Optimal Anonymization for *l*+-Diversity. In: Proceedings of the International Conference on Data Engineering, IEEE ICDE 2010 (2010)
28. Lunacek, M., Whitley, D., Ray, I.: A Crossover Operator for the *k*-Anonymity Problem. In: Proceedings of the GECCO Conference, pp. 1713–1720 (2006)

29. Machanavajjhala, A., Gehrke, J., Kifer, D.: *L*-Diversity: Privacy beyond *K*-Anonymity. In: Proceedings of the International Conference on Data Engineering (IEEE ICDE 2006), p. 24 (2006)
30. Miller, J., Campan, A., Truta, T.M.: Constrained *K*-Anonymity: Privacy with Generalization Boundaries. In: Proceedings of the Workshop on Practical Preserving Data Mining, with SIAM SDM 2008 (2008)
31. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases. UC Irvine (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
32. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, Los Altos (1993)
33. Samarati, P.: Protecting Respondents Identities in Microdata Release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
34. Schultz, S., Romacker, M., Faggioli, G., Hahn, U.: From Knowledge Import to Knowledge Finishing: Automatic Acquisition and Semi-Automatic Refinement of Medical Knowledge. In: Proceedings of the Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (1999)
35. Sweeney, L.: *k*-Anonymity: A Model for Protecting Privacy. International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems 10(5), 557–570 (2002)
36. Sweeney, L.: Achieving *k*-Anonymity Privacy Protection Using Generalization and Suppression. International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems 10(5), 571–588 (2002)
37. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison Wesley, Reading (2005)
38. Truta, T.M., Bindu, V.: Privacy Protection: *P*-Sensitive *K*-Anonymity Property. In: Proceedings of the Workshop on Privacy Data Management, with ICDE 2006, p. 94 (2006)
39. Wang, P.: Personalized Anonymity Algorithm Using Clustering Techniques. Journal of Computational Information Systems 7(3), 924–931 (2011)
40. Wei, Q., Lu, Y., Lou, Q.: (τ, λ)-Uniqueness: Anonymity Management for Data Publication. In: Proceedings of the IEEE International Conference on Computer and Information Science (2008)
41. Willemborg, L., Waal, T.: Elements of Statistical Disclosure Control. Springer, Heidelberg (2001)
42. Wong, R.C.W., Li, J., Fu, A.W.C., Wang, K.: (α, k)-Anonymity: An Enhanced k-Anonymity Model for Privacy-Preserving Data Publishing. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD 2006), pp. 754–759 (2006)

LORA: Link Obfuscation by Randomization in Graphs

Qian Xiao[1], Zhengkui Wang[1], and Kian-Lee Tan[1,2]

[1] NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore, Singapore
[2] School of Computing, National University of Singapore, Singapore

Abstract. In this paper, we propose a randomization scheme, LORA (Link Obfuscation by Randomization), to obfuscate edge existence in graphs. Specifically, we extract the source graph's hierarchical random graph model and reconstruct the released graph randomly with this model. We show that the released graph can preserve critical graph statistical properties even after a large number of edges have been replaced. To measure the effectiveness of our scheme, we introduce the notion of *link entropy* to quantify its privacy-preserving strength wrt the existence of edges.

Keywords: Link Obfuscation, Hierarchical Random Graph, Link Entropy.

1 Introduction

A graph is an effective structure that has been used to represent network data, where the nodes capture the entities and the edges reflect the relationships between the entities. For example, in social network applications such as facebook (facebook.com), a graph captures the friendships (edges) between individuals (nodes). By analyzing the graph, researchers can acquire interesting knowledge, such as how communities are formed and evolved, how diseases are transmitted and spread, and so on. However, the data associated with the graph also contain sensitive information, such as individual medical records in community health databases, personal friendship in social networks and private legislative collaborations among congressional representatives in government collaboration network. As such, it is critical to protect the privacy of personal information of released graphs.

While there has been numerous attempts to protect the privacy of personal information in released graphs, these methods are still vulnerable to various types of attacks [10,16,17,3]. Backstrom et al.[17] showed that, with very limited background knowledge, a great amount of nodes can be easily re-identified even after sanitizing the node's identity information such as social ID and name. More recently, Liu et al. [10] reported that node degree can be used as a quasi-identifier to re-identify node's identity in the graph. Zhou et al. also claimed that local subgraph knowledge such as a node's neighborhood can be easily retrieved by

© Springer-Verlag Berlin Heidelberg 2011

attackers. By matching the structure of the victim node's subgraph, attackers can trace and find the victim node [16]. Hay et al. also pointed out that hubs, as the fingerprints of graphs, are often uniquely identifiable. In fact, the popularity of social networks in recent years and the availability of powerful web crawling techniques have made personal information more easily accessible. Therefore, it is almost impossible to foresee an attacker's background knowledge in advance. Meanwhile, it is also unrealistic to make any assumptions on the constraints of an attacker's ability to collect such knowledge. As such, it is challenging to preserve privacy on graphs. This has prompted researchers to develop robust network/graph data protection techniques.

Existing works on preserving privacy of graphs fall into two main theoretical privacy models: k-anonymity-based model [7,8,9,10] and randomization model [3,1]. Under the k-anonymity-based model, a source graph is manipulated so that it has at least k corresponding entities satisfying the same structural knowledge. However, these methods are designed to be robust to certain specific attacks, e.g., k-degree [10] and k-automorphism [7] anonymization schemes protect the privacy of node degrees. Moreover, these works typically assume attackers' limited background knowledge. In addition, graph modification is restricted as the released graphs need to respect some symmetric properties in order for k candidates to share certain properties.

On the other hand, in randomization models [1,4,5,6], the released graph is picked from a set of graphs generated from a random perturbation of the source graph (through edge addition, deletion, swap or flip). Such an approach offers more freedom in "shaping" the released graph, i.e., no additional properties are intentionally injected. More importantly, an attacker's background knowledge would become unstable because of the random process. For example, by allowing random insertion and deletion of edges, an attacker is no longer 100% certain of an edge's existence. Moreover, randomization techniques are typically designed independent of any specific attacks, and hence are robust to a wider range of attacks. However, uncontrolled random purturbation means the space from which the released graph is picked is effectively "unbounded", making it difficult to preserve the source graph's structure. For example, if we allow only edge deletion, since edges are arbitrarily selected for deletion, important ties in a graph, such as bridge edges, may be eliminated resulting in a partitioned graph.

In this paper, we advocate and focus on randomization techniques. Our goal is to ensure that the released graph is privacy preserving, and yet useful for a wide range of applications. In particular, for the latter, the released graph should be "similar" to the source graph in terms of most properties (e.g., degree distribution, shortest path length and clustering coefficient). This raises three questions:

1. How to randomize a source graph so that the resultant released graph is still similar to it?
2. How to provide a measurement of shared information between the source and released graphs, to indicate the utility of the released graph ? Conversely, the measurement reflects the information loss due to randomization.

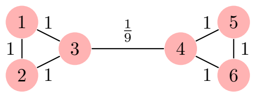
3. How to quantify the effectiveness of the randomized technique (and randomized graph) wrt privacy preservation? In other words, what is an appropriate measurable definition of privacy on graph?

From existing works, we can see much effort to address the first question above. In [4], the proposed approach restrains the changes in the random graphs' spectra to provide rough bounds of the random graph distribution. Another approach adopts the Metropolis-Hastings algorithm (specifically, the Markov Chain Monte Carlo method) to sample graphs with feature constraints [6,5]. This approach can preserve several graph statistical summaries, such as degree distribution, average clustering coefficient and average path length. However, since statistical summaries typically provide descriptions of a graph from different perspectives, but do not directly determine the graph structure, it is hard to quantify information lost since other graph features are not intentionally preserved. It is also not easy to evaluate its effectiveness wrt privacy preservation. In these works, the popular privacy measurement adopted merely relies on the different numbers of edges between the two graphs [5].

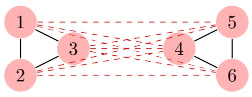
In this paper, we propose a randomization scheme, LORA (Link Obfuscation by Randomization), to generate a synthetic graph (from a source graph) that preserves the *link* (i.e., the extent of two node's relationship) while blurring the existence of an edge. In our context, *link* refers to the relation between two nodes. It is a virtual connection relationship, and is not necessarily a real edge that physically exists in the graph. We use the concept *link probability* as a quantity to measure the strength of *link*.

Next, we explain how LORA addresses the 3 questions that we raised. Firstly, we adopt the hierarchical random graph (HRG) model [2] to estimate each link probability in the source graph. The HRG model is a generic model that can capture assorted statistical properties of graphs. Based on the HRG model, we can randomly generate graphs that are similar to the source graph wrt statistical properties (i.e., dealing with the first challenge). Next, by reconstructing statistically similar graphs that preserve the source graph's HRG structure, we can select one to be released. In the ideal scenario, the released graph and source graph would share exactly the same HRG structure (i.e., addressing the second challenge).

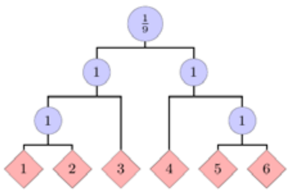
Third, to investigate how our method can preserve link privacy and how to quantify its strength, we introduce the notion of *link entropy*. Entropy has been widely used to measure the uncertainty of random variable in information theory. We will show that entropy is also appropriate in our scheme in terms of clarification and simplicity, compared to posterior belief that is used in previous works. Instead of analysing privacy with node's entropy [1], we define entropy based on links to theoretically quantify the effectiveness (wrt privary preservation) of our randomization scheme. As an attempt to address the third challenge, we will show how to derive the entropy for each individual link and then the composition of entropy of a set of links. We specifically define the notion of entropy of a node's egocentric network, which is an entropy ensemble and quantifies our scheme's privacy-preserving strength towards egocentric subgraphs. We will show how



(a) Source Graph



(b) Graph Obfuscation



(c) Dendrogram

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	1	1	1	0.11	0.11	0.11
v_2	1	1	1	0.11	0.11	0.11
v_3	1	1	1	0.11	0.11	0.11
v_4	0.11	0.11	0.11	1	1	1
v_5	0.11	0.11	0.11	1	1	1
v_6	0.11	0.11	0.11	1	1	1

(d) Link Probability Matrix

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	0	0	0.50	0.50	0.50
v_2	0	0	0	0.50	0.50	0.50
v_3	0	0	0	0.50	0.50	0.50
v_4	0.50	0.50	0.50	0	0	0
v_5	0.50	0.50	0.50	0	0	0
v_6	0.50	0.50	0.50	0	0	0

(e) Link Entropy Matrix

Fig. 1. An example of graph's hierarchical random graph(HRG) model. **c** is the dendrogram representation of graph **a**'s optimal HRG model. Dashed lines in **b** show the equivalent links of edge (3,4) in **a** according to the dendrogram **c**. **d** and **e** are **c**'s corresponding link probability matrix and link entropy matrix.

entropy quantifies an attacker's uncertainty accurately and clearly towards an egocentric network.

The rest of this paper is organized as follows. In Section 2, we provide some preliminaries. Section 3 gives an overview of our proposed LORA, and Section 4 presents the technical details of LORA. In Section 5, we analyze the privacy of our proposed LORA. Section 6 presents results of experimental studies. In Section 7, we review some related works. Finally, we conclude this paper in Section 8.

2 Preliminaries

2.1 Graph Notation

A graph $G(n, m) = (V, E)$ is a set of vertices V connected with a set of edges E, where $|V| = n$ and $|E| = m$. Let $A = (a_{ij})_{n \times n}$ be its adjacency matrix, where $a_{ij} = 1$ if node i and j are connected and $a_{ij} = 0$ otherwise. $\tilde{G}(\tilde{n}, \tilde{m}) = (\tilde{V}, \tilde{E})$ is the released graph reconstructed by randomization.

2.2 Hierarchical Random Graph and Its Dendrogram Representation

A graph often exhibits a hierarchical organization. Vertices can be clustered into subgraphs, each of which can be further subdivided into smaller subgraphs, and so forth over multiple scales. The hierarchical random graph (HRG) model is a tool to explicitly describe such hierarchical organization at all scales for a graph. According to Clauset's experiments [2], the graphs "resampled" with HRG can match the statistical properties of the source graphs closely, including degree distributions, clustering coefficients, and distributions of shortest path lengths.

HRG can be represented with a dendrogram tree. Let D be the corresponding dendrogram of the source graph G. A dendrogram D is a binary tree with n leaf nodes corresponding to the n vertices of G [2,11]. Let r be the lowest common ancestor of the two nodes in D. Let L_r and R_r be the number of leaf nodes in the two subtrees of r, respectively. And let E_r denote the number of edges in graph G whose endpoints correspond to leaf nodes from each of two subtrees of r in D. Each inner node r of one dendrogram is associated with a probability p_r, which describes the probability of connections between two groups of leaf nodes in the two subtrees of r. In HRG, we use $E_r/(L_r \cdot R_r)$ to be the maximum likelihood estimator of p_r. The likelihood of one HRG model for a given graph measures how plausible this HRG is to represent the graph. It can be calculated as follows:

$$\mathcal{L}(D, \{p_r\}) = \prod_{r \in D} p_r^{E_r} (1 - p_r)^{L_r R_r - E_r} \tag{1}$$

As an example, Fig. 1(c) is a dendrogram representation of the graph in Fig. 1(a). At the top scale of dendrogram in Fig. 1(c), vertices in graph G are divided into two groups from the root r in dendrogram D, corresponding to the leaf set {1,2,3} in the left subtree of D and leaf set {4,5,6} in the right subtree, respectively. Each group has 3 leaf nodes, so $L_r = R_r = 3$. Since only one edge (3,4) exists in the graph Fig. 1(a), E_r should be one. And the probability p_r of connections between two groups can be estimated as $E_r/(L_r \cdot R_r) = \frac{1}{9}$.

The optimal HRG that fits an orginal graph can be determined using the Markov Chain Monte Carlo method (MCMC). In practice, most real world networks will have many plausible hierarchical representations of roughly equal likelihood even after reaching equilibrium, which may slightly differ in arrangement of tree's branches. We sample dendrograms at regular intervals and calculate the mean probability p_{ij} for each pair of vertices i, j. In our analysis, we assume the dendrogram derived by MCMC is always the ideal one that fits the source data best. For instance, we assume Fig. 1(c) is Fig. 1(a)'s ideal dendrogram. From Fig. 1(c), we note that all p_{ij} can be quantified with $E_r/(L_r \cdot R_r)$ as shown in the probability matrix in Fig. 1(d).

2.3 Entropy

Entropy measures the uncertainty regarding the value of one random variable in information theory. The less probable the outcome of one random variable X is, the greater its entropy is.

A random variable X has a probability p to render an outcome x, and a probability $1-p$ to generate another alternative outcome x'. The uncertainty of an outcome of this random variable X is defined as a binary entropy function,

$$H_2(X) = p \log_2 \frac{1}{p} + (1-p) \log_2 \frac{1}{1-p}, \tag{2}$$

with the convention that $0 \times \log 1/0 = 0$.

An ensemble random variable X, where the outcome x is the value of X, can take on one of a set of possible values, $\mathcal{C}_X = \{c_1, c_2 ..., c_I\}$. $\mathcal{C}_X$ has probabilities $\mathcal{P}_X = \{p_1, p_2, ..., p_I\}$. The entropy of the ensemble variable X is,

$$H(X) = H(p_1, p_2, ..., p_I) = \sum_{i=1}^{I} p_i \log_2 \frac{1}{p_i}, \tag{3}$$

Entropy has additive properties for independent variables. That is, if variable X and Y are independent, the entropy of the outcome x, y satisfies,

$$H(X, Y) = H(X) + H(Y) \tag{4}$$

In addition, $H(X) \geq 0$ with equality iff $p_i = 1$ for one i.

3 LORA: The Big Picture

Our proposed LORA framework consists of two main steps: (1) Find a HRG model that best fits the source graph; (2) Based on the HRG, reconstruct a new graph by random link sampling. Algorithm 1 outlines the LORA framework.

We firstly introduce two critical concepts: *link* and *link probability*. In our context, the term *link* refers to the relation between two nodes. The term *link probability* is one quantity to measure the strength of *link*. These two concepts are appropriate to depict such a scenario: A pair of nodes, although not directly connected in the source graph, may still have a weak relation (*link*) if the two shares many common neighbours. Due to these common neighbours, a promising connection may appear in the future. For example, in social networks, friends of friends would more likely become friends soon. To distinguish, we use the term "edge" if there is a direct connection between two nodes in a graph.

The role of each link differs in its impact on topology. For instance, in Fig. 1(a), links (3,4) and (1,6) exhibit the same topological effect, i.e., they are exchangeable. Thus, we can replace edge (3,4) in Fig. 1(a) with link (1,6) without destroying any topological structure (see Fig. 2). Moreover, the role of vertex is fully determined by the links incident on the vertex. As Fig. 1(a) shows, vertices 3 and 4 have the same roles, and so are vertices 1 and 2.

In order to estimate such *link probabilities* in the source graph, we adopt the hierarchical random graph (HRG) model in LORA. More specifically, we use the Markov Chain Monte Carlo (MCMC) method to find the HRG model that best fits the source graph. We choose HRG as our model because it is a generic model, which describes a graph's structure in detail, including all the probabilities of a connection between any two vertices in the graph. Here we extend the concept of this probability to be *link probability* in order to quantify our "link" notion. In addition, in [2], Clauset claimed that the HRG model can capture assorted statistical properties of a graph. It is also shown that hierarchy is a central organizing principle of networks [2]. In contrast to simple clustering, HRG describes organization at all scales in a network. Intuitively, once the change in the HRG model can be restricted, it would naturally bound the distribution of regenerated random graphs, since the HRG models they hold are similar.

Now, we begin to describe the two steps in LORA. At the first step of LORA, we determine the best HRG model of the source graph by running MCMC sampling algorithm until equilibrium and represent it as a dendrogram tree D (See Algorithm 1, line 1). Leaf nodes in D correspond to vertices in the graph. Each non-leaf internal node is associated with two communities(i.e. two sets of leaf nodes) induced by its left and right subtrees (line 3,4). Ideally, links across these two communities are viewed as approximately equivalent and exchangeable relationships in terms of the inter-community association strength. We denote such a group of equivalent links as one link equivalent class (line 5).

Secondly, in the reconstruction step (line 6,7), we replace true edges in the source graph with their equivalent links in link equivalent classes. In order to maintain the same inter-community association strength, we randomly pick the same number of links in the link equivalent class to substitute the true edges observed in the source graph and let the chosen links be the new edges in the released graph (lines 7). Note that obfuscation also comes simultaneously from the inherent random process. In the privacy analysis part, we would introduce the concept of "*link entropy*" to assess the degree of privacy brought by link obfuscation.

```
Data: A simple source Graph $G(V, E)$
Result: A reconstructed random Graph $\tilde{G}(\tilde{V}, \tilde{E})$ for release, where $\tilde{V} \subseteq V$
1 Dendrogram $D \longleftarrow$ fitHRG($G$)
2 foreach nonleaf node $r$ in $D$ do
3     $V_{left}(r) \leftarrow$ findLeafVertices(left subtree of $r$)
4     $V_{right}(r) \leftarrow$ findLeafVertices(right subtree of $r$)
5     link equivalent class $L(r) \leftarrow V_{left}(r) \times V_{right}(r)$
6     $E(r) \leftarrow$ the number of observed edges $\in V_{left}(r) \times V_{right}(r)$ in $G$
7     randomly pick up $E(r)$ links in current equivalent class $L(r)$ to be new edges
      in $\tilde{G}$
8 end
```

Algorithm 1. The LORA Framework

4 Link Obfuscation by Randomization with HRG

4.1 Link Equivalence Class

Given a network $G(n, m) = (V, E)$ and its HRG dendrogram tree D, links bridging the nodes in the left and right subtrees are in the same equivalence class with respect to their topological roles. Consider the graph G in Fig. 1(a) and its dendrogram in Fig. 1(c). At the top level of G's dendrogram tree, the dendrogram divides into two subtrees, which induces two separate leaf sets - left subtree leaf set {1,2,3} and right subtree leaf set {4,5,6}. All the possible cross links bridging these two leaf sets consist of one link equivalence class. In this case, as shown by the dash lines in Fig. 1(b), links (1,4), (2,4), (3,4), (1,5), (2,5), (3,5), (1,6), (2,6), (3,6) are 9 pairs of links in one equivalence class. Let L_r and R_r be the sizes of the left and right leaf sets respectively (in our example, $L_r = 3$, $R_r = 3$). Let E_r be the number of edges in the source graph G linking the two sets (in Fig. 1(a), there is only one edge (3,4), so $E_r = 1$). We can then estimate the link probability of this link equivalence class as $E_r/(L_r \cdot R_r) = 1/(3 \cdot 3) = 1/9$. This probability indicates the connection strength between the nodes in the two leaf groups. In this sense, we obfuscate the existence of connections of nodes, by turning real edges into virtual probabilistic links.

Note that, ideally, if links in one equivalent class share exactly the same topological roles, the new generated graph should also share exactly the same dendrogram as the source graph. However, this is not usually the case. Very often, the equivalent link class derived through HRG is just a group of approximately topological similar links. Besides, for computation efficiency, we have adopted a local optimal HRG right after the MCMC runs into local equilibrium state. For these reasons, links in one link equivalent class we derive are actually just approximately equivalent. As such, the released graph's optimal HRG is very likely changed a bit compared to the optimal HRG of the source graph. In this paper's analysis, we assume the released graph shares exactly the same global optimal dendrogram of the source graph. Note that, from a privacy's perspective, it is intuitive that, if the dendrograms of the source and released graphs are not the same, it would be much harder to infer the source graph from the released graph. Therefore, this assumption is biased against LORA in terms of its privacy strength. Since the statistical relationship of each link in the graphs are fully preserved in this ideal scenario, such released graphs would share the same inherent statistical properties with the source graph. Essentially, all information that the released and source graphs share is the dendrogram D in the ideal scenario.

4.2 Link Replacement

Now we explain the link replacement procedure with HRG model during graph reconstruction. We reconstruct random graph by a series of link replacement procedures, where each inner node in HRG corresponds to one link replacement procedure. Consider one inner node r in dendrogram D. There are E_r real edges in G bridging the two leaf node sets in the left and right subtrees in D. In order to maintain the same connection strength between the two leaf node sets in the released graph $\tilde{G}$, we randomly pick E_r links in an inner node r's link equivalent class to replace the E_r real edges. The whole reconstruction process is done through $n-1$ independent link replacement procedures, corresponding to $n-1$ inner nodes in HRG. Referring to our running example again, consider the root node in the dendrogram, since there is only one real edge (3,4), we need to find a link to replace it. Fig. 2 shows the released graph after link (1, 6) replaces edge (3,4).

4.3 Hide Weak Ties and Retain Strong Ties

Essentially, HRG model tends to exchange weak ties(yet true edge) in the source graph with links that are not connected yet. See the instance of graph in Fig. 1(a) and its perturbed graph in Fig. 2. By "weak ties", we mean edges that are bridges to link two graph components, such as edge a_{34} in Fig. 1(a). As real world graphs are typically sparse, weak ties are not uncommon. In fact, they are important channels between many clustered groups and hold important roles in shaping the entire graph structure. In pure random edge deletion scheme, such weak ties may be removed, which will severely undermine the source graph's structure. However, under our LORA scheme, link obfuscation is employed to substitute such weak ties with fake (non-existent) ties within the same equivalence class. In this way, large amount of changes can be operated on the graph while preserving the skeleton of source graph as well as the clique-like components.

Using Fig. 2 as illustration, we associate (small) link probabilities with weak links, that is, links between node set {1,2,3} and {4,5,6}, which give much freedom to perturb the source graph to obfuscate links in the released graph. In this case, nodes 1, 2, 3, which are rooted in the same inner node, have exactly the same link relationship towards all the other nodes. Therefore, they are interchangeable. In Fig. 2, the skeleton (the dashed line and dashed circles) of the perturbed graph (of the graph in Fig. 1(a)) remains the same as the source graph. For complete components, namely, clique, link probability obfuscation usually would preserve them fully since they link with each other closely. This makes sense since cliques are obvious features in graphs, one cannot perturb one complete graph without destroying its property of complete graph. To improve the privacy of nodes in a clique, an alternative way is to compress cliques into one super node. That is, in our context, to generalize all the leaf nodes induced from one subtree in the dendrogram to one supernode. With an ideal dendrogram, it is an effective way to compress subcomponents of graph with minimal disruptions to the remaining structure.

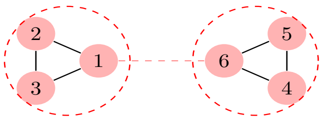


Fig. 2. Perturbed Graph

5 Privacy Analysis

In this paper, we use entropy as the privacy criterion to quantify the strength of link obfuscation method in preserving link privacy.

As an analogy of binary entropy function, each link a_{ij} with link probability p_{ij} has binary link entropy,

$$H(a_{ij}) = p_{ij} \log_2 \frac{1}{p_{ij}} + (1 - p_{ij}) \log_2 \frac{1}{1 - p_{ij}} \tag{5}$$

Link entropy quantifies the degree of uncertainty of the existence of a edge, i.e., whether nodes i and j are connected or not in source graph. In general, a larger entropy value is preferred. For example, the table in Fig. 1(e) is a matrix consisting of all link entropies, which is derived by the probability matrix in Fig. 1(d) for the graph in Fig. 1(a). As shown in Fig. 1(e), links between node set {1,2,3} and set {4,5,6} have entropy 0.50, indicating the uncertainty level of the real state of the links measured in entropy.

5.1 The Joint Link Entropy

It is not uncommon for attackers to attempt to identify a set of links, e.g. check the egocentric network of one node (i.e., all edges incident to that node), search for subgraphs and so on. To this end, we then formalize the joint link entropy to quantify the degree of uncertainty in these scenarios.

Joint Entropy of Dependent Links. For the links associated with the same inner node r in a dendrogram D, they are dependent (or relevant) random variables. Consider observing the outcome of k ($k \leq L_r \cdot R_r$) dependent links whose endpoints are rooted at the same lowest common ancestor r. We use a joint ensemble variable $X_r = a^r_{i_1 j_1} a^r_{i_2 j_2} ... a^r_{i_k j_k}$ to represent the ensemble of such k link variables under observation. The ensemble variable X_r can take on one of a set of possible ensemble outcomes, x^s_r, which consists of the outcome of each link variable $a^r_{ij} \in \mathcal{C}_{a_{ij}} = \{0, 1\}$. Here $a^r_{ij} = 1$ denotes link a^r_{ij} is chosen during link replacement; otherwise, for non-chosen links, $a^r_{ij} = 0$. Each specific outcome x^s_r has a probability p_s. In the context of link replacement, p_s refers to the possibility one specific outcome x^s_r of the k relevant links (chosen or unchosen) appears after link replacement. The link ensemble X_r has a probability distribution $\mathcal{P}_X = p_1, p_2, ..., p_S$, where p_s is the joint of probability for the ensemble outcome.

As one example, we consider the outcome values(i.e. the possible outcomes) of links $X_r = a_{14}a_{24}a_{34}$ in Fig. 1(a). X_r can take on one of 4 ordered sequence outcomes, that is, no link selected from $\{a_{14}, a_{24}, a_{34}\}$(outcome "000"); a_{34} selected(outcome "001");a_{24} selected(outcome "010") and a_{14} selected(outcome "100"). Consider the calculation of $p_s(000)$, the probability that the outcome value of link set $a_{14}a_{24}a_{34}$ is 000 after link replacement. First of all, after the link replacement regarding inner node r, there are $\binom{L_r \cdot R_r}{E_r}$ types of outcomes for the whole link equivalent class. Among all the outcomes, we count the number of outcomes where link $\{a_{14}a_{24}a_{34}\}$ is 000. Sequence 000 indicates that in the observed link set $\{a_{14}, a_{24}, a_{34}\}$, none of links is chosen. Let l denote the number of links chosen from the k relevant links. Hence, in sequence 000, $l = 0$. The observed link set size k is 3 here. In order to replace 1 ($E_r = 1$) original edge in G, another 1 ($E_r - l = 1$) link needs to be drawn from the rest links in the inner node r's link equivalence class($L_r \cdot R_r - k$ links). There are $\binom{L_r \cdot R_r - k}{E_r - l}$ types of ensemble outcomes for drawing $E_r - l$ links from the rest links. Hence $p_s(000)$ is $\binom{3 \cdot 3 - 3}{1 - 0} / \binom{3 \cdot 3}{1} = 6/9$. In the following, we give the generalized formula of p_s:

$$p_s = \frac{\binom{L_r \cdot R_r - k}{E_r - l}}{\binom{L_r \cdot R_r}{E_r}}, \tag{6}$$

where l is the number of links drawn from the observed k relevant links.

The joint entropy of dependent links is defined as,

$$\begin{aligned} H(X_r) &= H(a^r_{i_1 j_1} a^r_{i_2 j_2} ... a^r_{i_k j_k}) \\ &= H(p_1, p_2, ..., p_S) = \sum_{s=1}^{S} p_s \log_2 \frac{1}{p_s}, \end{aligned} \tag{7}$$

which measures the degree of attacker's uncertainty regarding a set of dependent links.

As illustration, we again consider the possible outcomes of links $X_r = a_{14} a_{24} a_{34}$ in Fig. 1(a). The space of X_r's possible worlds consists of 4 binary sequences $\{000, 001, 010, 100\}$ with the probability distribution $\{6/9, 1/9, 1/9, 1/9\}$. Note that the possible outcomes are dependently distributed, yet not identical. The corresponding joint entropy of X_r is 1.45.

Joint Entropy of Independent Links. For the links associated with different inner nodes, they are independent random variables. The joint ensemble of independent links is the sum of the link entropy of each link, i.e.,

$$H(X) = H(a^{r_1}_{i_1 j_1} a^{r_2}_{i_2 j_2} ... a^{r_k}_{i_k j_k}) = \sum_{h=1}^{k} H(a^{r_h}_{i_h j_h}), \tag{8}$$

Joint Entropy of Arbitrary Links. Given a set of arbitrary links, we separate them into two categories: independent links and dependent links. Essentially we arrange the links in different groups according to the inner node in the dendrogram they associate to. Links in the same group are dependent. Otherwise, they are independent. The joint entropy of arbitrary links is the sum of the joint entropy of each group, which is given by the following equation,

$$H(X_{r_1, r_2 ... r_t}) = \sum_{k=1}^{t} H(X_{r_k}) \tag{9}$$

In order to provide a more meaningful measure of each vertex's privacy, we next define the notation of the entropy of a vertex's egocentric network. The egocentric network is one smallest subgraph centered on each node. The egocentric network entropy is an ensemble entropy regarding all the immediate links associated with the node,

Definition 1. *(Joint entropy of vertex i's egocentric network) $H(v_i)$ is the entropy of the joint link entropy $H(a_{ij_1}a_{ij_2}...a_{ij_{n-1}})$ which includes all links incident to vertex i.*

This definition quantifies an attacker's uncertainty towards the composition of one vertex's egocentric network.

Traditionally, for randomization schemes, the posterior belief is used to measure an attacker's uncertainty [4][3]. Here we use entropy rather than the posterior belief for clarifying the ensemble uncertainty of possible worlds. Consider a link with probability p. For an attacker, there are two scenarios - $a_{ij} = 0$ or $a_{ij} = 1$. Rather than specifying that the attacker has posterior belief p for $a_{ij} = 1$ and posterior belief $1 - p$ for $a_{ij} = 0$, we use $H(a_{ij})$ to evaluate the attacker's uncertainty of this link random variable as a whole. $H(a_{ij})$ reflects the extent to which an attacker is unsure of a_{ij}'s real outcome in all its possible worlds. Note that the possible worlds are not always evenly distributed. Entropy describes the extent of obfuscation compactly instead of specifying several probabilities of each possible world. This is particular convenient in more intricate scenarios, especially in the case of joint entropy.

Essentially, the released graph conveys the same amount of information contained in the dendrogram, which indicates that the most amount of information attackers can infer from one released graph is just the dendrogram, by using MCMC to learn from the released graph. Note that each link replacement procedure associated with one inner node in the dendrogram cannot be directly learned, since it operates as a non-deterministic mapping function. Information transferred after link obfuscation is inherently blurred according to the HRG model.

5.2 Link Obfuscation Vs Node Obfuscation

In [1], Bonchi et. al. claimed that entropy-based quantification of anonymity level is more adequate than quantification based on posteriori belief probabilities. Our approach is similar in spirit to their work, but differs crucially in the quantity under measurement. Rather than defining the quantity of node identity anonymity level directly, we consider an entropy quantification for links. Bonchi's work is mainly concerned with re-identification of node identity, while our work attempts to address re-identification of links. Moreover, in [1], node candidates are the vertices in the released graph $\tilde{G}$. But in our scheme, link candidates are the imaginary possible worlds during obfuscation scheme.

Furthermore, we show that link entropy distinguishes the uncertainty of links in different distribution of possible worlds under randomization scheme. As illustration, we consider the following two situations:

(1) $p(a_{12} = 0, a_{13} = 1) = \frac{1}{2}$,
$p(a_{12} = 1, a_{13} = 0) = \frac{1}{2}$,
$p(a_{12} = 0, a_{13} = 0) = 0$,
$p(a_{12} = 1, a_{13} = 1) = 0$.

(2) $p(a_{12} = 0, a_{13} = 1) = \frac{1}{2}$,
$p(a_{12} = 1, a_{13} = 0) = \frac{1}{6}$,
$p(a_{12} = 0, a_{13} = 0) = \frac{1}{6}$,
$p(a_{12} = 1, a_{13} = 1) = \frac{1}{6}$.

We note that with the probabilities listed above, it is hard to evaluate which of the two cases brings more uncertainty. We next show how link entropy can distinguish the extent of these two cases' uncertainty. According to Equation 7, in case 1, the joint link entropy is $\log_2 2$, but it is $\log_2 2\sqrt{3}$ in case 2. Although an attacker's greatest confidence about the state of links in released graph is both 1/2 in two the cases, the attacker needs more effort to cross out the more uncertain possible worlds in the second case. Under LORA, during link replacement, the existence of a weak tie in graph is blurred since the link probability is effectively being spread among the fake candidate links in the equivalence class. Thus, the uncertainty of possible worlds of links is increased.

Our scheme is specially designed for link privacy in the first place. But more importantly, the uncertainty of links would directly undermine the structural knowledge that the attackers can hold in any attacks. This is because links, the smallest atomic elements in graph, are the foundations of all the structure knowledge attackers can hold in a simple graph.

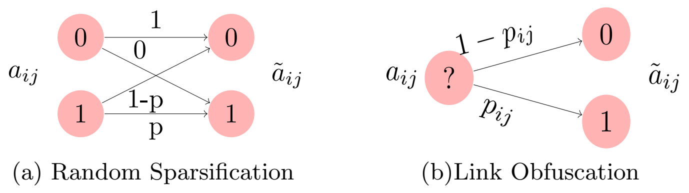


Fig. 3. Link Obfuscation vs Random Sparsification

5.3 Randomization by Link Obfuscation vs Edge Addition/Deletion

Unlike randomization schemes in [3,4,6,1], link probability obfuscation is a sophisticated method based on the source graph's characteristics. We use Fig. 3 to illustrate the difference between random sparsification [1] and link obfuscation. For the pure random sparsification, links are perturbed in a way similar to a coin flipping game, where the coins are the same and independent. As it turns out in Fig. 3(a), every a_{ij} is associated with the same parameter p in the procedure of perturbation. Each a_{ij} "flips" like a coin in the same way. Conversely, in LORA (see in Fig. 3(b)), each a_{ij} owns its specific perturbation parameter p_{ij}. Each inner node in the dendrogram is associated with one independent link replacement procedure. During each procedure, links in the link equivalence class are all related. This means more dedicate modifications are allowed on the source graph.

From entropy's perspective, in the former scheme, all $a_{ij} = 0$ will retain its state in the released graph. Therefore, the entropy $H(a_{ij}|\tilde{a}_{ij} = 1) = p(a_{ij} =$

$0|\tilde{a}_{ij} = 1) \log \frac{1}{p(a_{ij}=0|\tilde{a}_{ij}=1)} + p(a_{ij} = 1|\tilde{a}_{ij} = 1) \log \frac{1}{p(a_{ij}=1|\tilde{a}_{ij}=1)} = 0 \cdot \log(1/0) + 1 \cdot \log(1/1) = 0$. This implies that an attacker can learn that $a_{ij} = 1$ if the observation is $\tilde{a}_{ij} = 1$ in the released graph. However, in the latter scheme, p_{ij} that a_{ij} associates with is not necessarily always 0 or 1. Hence, if $p_{ij} \neq 0$ or $p_{ij} \neq 1$, it is almost not learnable from the obfuscation procedure. In other words, it is difficult to infer the true state in the source graph with full confidence.

6 Experimental Studies

In this section, we report results of two experimental studies to evaluate the effectiveness of LORA. We report results from two real datasets, `adjnoun` [13] and `celegans` [14]. `adjnoun` is an undirected graph of common adjective and noun adjacencies for the novel "David Copperfield" by Charles Dickens. Edges connect any pair of words that occur in adjacent position in the text of the book. `celegans` is a biological network representing the neural network of C. Elegans.

6.1 Graph Statistics

We measure the following statistics: graph diameter (i.e., the maximum distance among pairs of vertices), histograms of degree distribution (i.e., the probability distribution of vertices degrees over the whole graph), shortest path lengths (i.e., the shortest path lengths between any two nodes in graph) and clustering coefficients (i.e., for each node, find the fraction of possible triangles that exist). We also measure the join entropy histogram of each vertex's egocentric network.

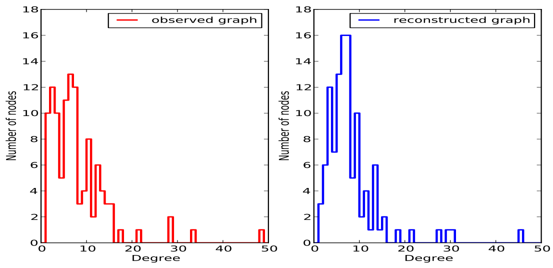
For the `adjnoun` dataset, the source graph has 112 vertices and 425 edges. In the released random graph, the number of vertices $\tilde{n}$ is 110 and the number of edges $\tilde{m}$ is 425. Only two vertices in source graph are lost (disconnected with the largest component of released graph) during link obfuscation. 301 original edges are substituted with previously non-connected links in the released graph.

The Source graph of `celegans` dataset has 297 vertices and 2148 undirected edges, whereas its released random graph has 288 vertices and 2148 undirected edges. 1640 original edges are replaced with previously non-connected links in the released graph.

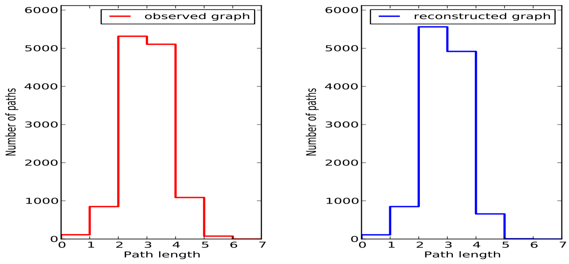
The results, shown in Fig. 4 and Fig. 5, are summarized in the following.

Graph Diameter: The diameters of the source and released `adjnoun` graphs are both 5. For the `celegans` dataset, both the source and released graphs' diameters are also equal to 5.

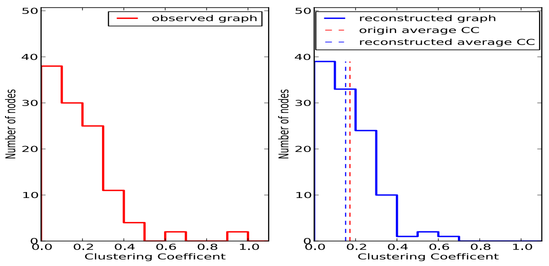
Degree Distribution: Fig. 4(a) shows that the degree distribution histogram of the source (left figure in red) and released (right figure in blue) graphs for the `adjnoun` dataset. We observe that the two histograms share similar profiles. Specifically, both histograms have "fat tails" in their distributions, containing some nodes with large degrees greater than 30. It is known that the network with the scale-free property is characterized by its fat-tail. The result for the `celegans` dataset, shown in Fig. 5(a), also demonstrates the same property.



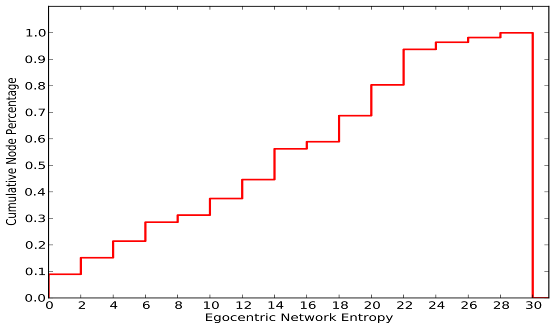
(a) adjnoun - Degree Distribution



(b) adjnoun - Shortest Path Length

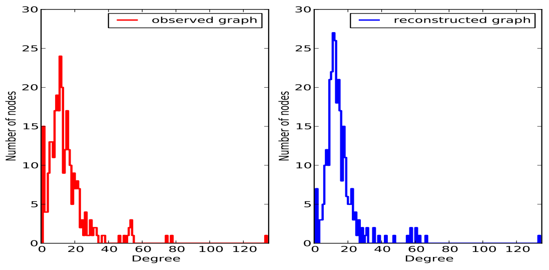


(c) adjnoun - Clustering Coefficients

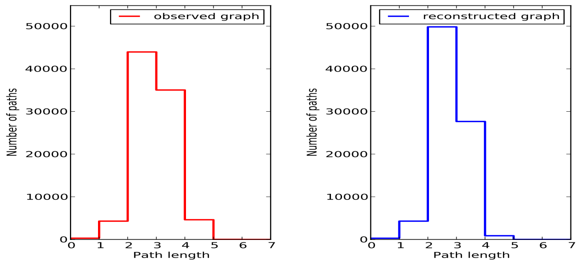


(d) adjnoun - Egocentric Entropy

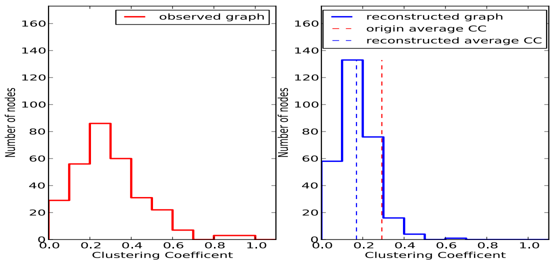
Fig. 4. Performance analysis on dataset `adjnoun`



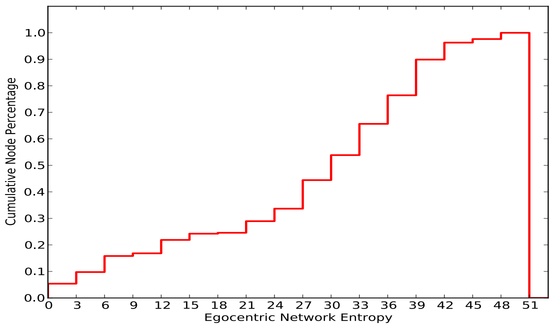
(a) celegans - Degree Distribution



(b) celegans - Shortest Length Path



(c) celegans - Clustering Coefficients



(d) celegans - Egocentric Entropy

Fig. 5. Performance analysis on dataset `celegans`

Shortest Path Lengths: The results of the shortest path length histograms for the two datasets are shown in Fig. 4(b) and Fig. 5(b) respectively. As shown, for both datasets, the released and source graphs share similar shortest path distribution.

Clustering Coefficients: From the result of the clustering coefficients histogram (see Fig. 4(c) for the `adjnoun` dataset and Fig. 5(c) for the `celegans` dataset), we observe again the similarity in the two histogram profiles. Moreover, for the `adjnoun` dataset, as shown in Fig. 4(c), the average clustering coefficients (vertical dashed lines in the right figure) are almost the same. For the `celegans` dataset, the average clustering coefficients are also close.

To summarize, we can see that link probability obfuscation can preserve graph features well.

6.2 Privacy Analysis

To understand the effectiveness of LORA, we report the vertex egocentric entropy in Fig. 4(d) and Fig. 5(d). We use Fig. 4(d) for explanation. Fig. 4(d) shows the cumulative histogram of egocentric entropy for all vertices in the source graph of dataset `adjnoun` according to the derived HRG model used to reconstruct the released graph. The horizontal axis specifies the egocentric entropy interval. The vertical axis specifies the percentage of nodes whose entropy is less than the right value of the corresponding entropy interval in the horizontal axis for this height. As an example, for the interval 10-12, about 38% of the nodes' entropy is less than 12; and 100% of the nodes' entropy is less than 30. Now, the more egocentric entropy one node holds, the more privacy it gets. The lowest value of entropy is zero, meaning there is no uncertainty regarding the node's egocentric network. Notice that the entropy we measure here is a subgraph's entropy, that is, each node's egocentric network. Fig.4(d) is the histogram of this type of entropy with regard to all the nodes' egocentri networks in the graph.

From Fig. 4(d), we note that less than 30% of the vertices' egocentric entropy is smaller than 8, which means, for at least 70% of the vertices' egocentric network, an attacker's uncertainty is above 8 in terms of entropy. Less than 10% of the vertices have egocentric entropy smaller than 2. Assume 2 is the entropy threshold δ, nodes with entropy value lower than this threshold are the vulnerable targets. In this case, attackers can believe their egocentric networks in the released graph are likely to be close to the source egocentric networks. However, it should be noted that for such nodes with low egocentric entropy can be further processed in order to improve their privacy. Because often link probabilities of such nodes are near 1, most likely to be the nodes in cliques. Clique-like nodes can be compressed into a super node, without significantly disrupting the rest graph structure. We leave this issue for further study.

7 Related Work

In recent works on randomization based scheme, Ying considered graph's spectra as an indicator to adjust released graph. Graph's spectra, consisting the set of graph eigenvalues, are known as quantities that characterize one graph. It is also conceivable that almost all graphs are uniquely determined by their spectra[15]. However, it is not clear how precisely edge deletion/addition would influence graph's spectral but only rough bound for just the two important eigenvalues among them, namely, the largest eigenvalue(λ_1) for the adjacency matrix A and the second eigenvalue(μ_2) for the Laplacian matrix L. In [5][6], Ying and Hanhijarvi considered approaches to generate synthetic graphs with Metropolis–Hastings algorithm(more specifically, Markov Chain Monte Carlo method). Essentially, these two methods extract important parameters of original graph, such as λ_1, μ_2, h(the harmonic mean of the shortest distance), and then use Markov Chain Monte Carlo method to sample the set of graph with same parameters as the original graph. To this perspective, they effectively bound the subspace of samples so that guarantee its released graph preserves one same feature as original graph. However, preserving one statistical summaries does not always guarantee preserving other summaries simultaneously according to their observation. In addition, they also observe that the attacker can utilize the same strategy to exploit the graph space, which will jeopardize approximately 20% true edges to expose, dependent on different parameters. It is not easy to analyze this risk theoretically since it varies according to different parameters, which hence renders precisely assessing the privacy-preserving strength of such strategy difficult. In [12], Mir considered one approach from the opposite perspective. Rather than sampling a particular distribution in graph space according to certain parameters, Mir proposed to learn from original graph with Leskovec et al.'s Kronecker graph model firstly and then generate one synthetic graph. Because Kronecker graph model theoretically captures some key features of realistic graphs[12], the synthetic graphs will generically share almost same features. However, in order to respect their graph differential privacy model, they consider to inject Laplacian noise on the parameters of their graph model before graph generation, which intentionally disorders the sample's distribution induced from true graph model a bit. In addition, their privacy model and sensitivity criteria consider just parameters of graph model, not on graph itself. It is not clear how exactly privacy criteria imposed on graph model parameters would exert privacy protection on released graph.

8 Conclusion

In this paper, we have proposed a randomization scheme, LORA, to preserve link privacy of network data publishing. LORA builds the HRG model of the source graph, and uses it to reconstruct a set of graphs that preserve the statistical graph properties of the source graph. The released graph is then selected from these graphs. We also introduced and argued that the *link entropy* concept is an appropriate measure of the uncertainty degree of links. Our experimental results showed that the released (reconstructed) graphs have acceptable link entropy while preserving statistical properties such as graph diameter, degree distribution, shortest path lengths and clustering coefficients.

Acknowledgement. Qian Xiao and Zhengkui Wang are supported by scholarships from NGS/NUS. Kian-Lee Tan is partially supported by a research grant R-252-000-433-305.

References

1. Bonchi, F., Gionis, A., Tassa, T.: Identity Obfuscation in Graphs Through the Information Theoretic Lens. In: IEEE International Conference on Data Engineering, pp. 924–935. IEEE Press, Hannover (2011)
2. Clauset, A., Moore, C., Newman, M.E.J.: Hierarchical structure and the prediction of missing links in networks. Nature 453, 98–101 (2008)
3. Hay, M., Miklau, G., Jensen, D., Weis, P., Srivastava, S.: Anonymizing social networks. University of Massachusetts Technical Report (2007)
4. Ying, X., Wu, X.: Randomizing social networks: A spectrum preserving approach. In: Proceedings of the 8th SIAM Conference on Data Mining, pp. 739–750 (2008)
5. Ying, X., Wu, X.: Graph generation with prescribed feature constraints. In: Proceedings of the 9th SIAM Conference on Data Mining, pp. 966–977 (2009)
6. Hanhijarvi, S., Garriga, G.C., Puolamaki, K.: Randomization techniques for graphs. In: Proceedings of the 9th SIAM Conference on Data Mining, pp. 780–791 (2009)
7. Zou, L., Chen, L., Ozsu, M.T.: K-automorphism: A general framework for privacy preserving network publication. VLDB, 946–957 (2009)
8. Wu, W., Xiao, Y., Wang, W., He, Z., Wang, Z.: k-symmetry model for identity anonymization in social networks. EDBT, 111–122 (2010)
9. Cheng, J., Fu, A.W.C., Liu, J.: K-isomorphism: privacy preserving network publication against structural attacks. SIGMOD, 459–470 (2010)
10. Liu, K., Terzi, E.: Towards identity anonymization on graphs. SIGMOD, 93–106 (2008)
11. Clauset, A., Moore, C., Newman, M.E.J.: Structural Inference of Hierarchies in Networks. CoRR, 1–13 (2006)
12. Mir, D.J., Wright, R.N.: A Differentially Private Graph Estimator. In: ICDM Worshops, pp. 122–129 (2009)
13. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E 74, 036104 (2006)
14. White, J.G., Southgate, E., Thomson, J.N., Brenner, S.: The Structure of the Nervous System of the Nematode Caenorhabditis elegans. Phil. Trans. R. Soc. London 314, 1–340 (1986)
15. Dam, E.R.V., Haemers, W.H.: Spectral Characterizations of Some Distance-Regular Graphs. Journal of Algebraic Combinatorics, 189–202 (2002)
16. Zhou, B., Pei, J.: Preserving Privacy in Social Networks Against Neighborhood Attacks. ICDE, 506–515 (2008)
17. Backstrom, L., Dwork, C., Kleinberg, J.: Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In: Proceedings of the 16th International Conference on World Wide Web, pp. 181–190 (2007)
18. Hay, M., Miklau, G., Jensen, D., Towsley, D., Weis, P.: Resisting structural re-identification in anonymized social networks. In: Proceedings of the VLDB Endowment, pp. 102–114 (2008)

A Comprehensive Framework for Secure Query Processing on Relational Data in the Cloud

Shiyuan Wang, Divyakant Agrawal, and Amr El Abbadi

Department of Computer Science, University of California at Santa Barbara
{sywang,agrawal,amr}@cs.ucsb.edu

Abstract. Data security in the cloud is a big concern that blocks the widespread use of the cloud for relational data management. First, to ensure data security, data confidentiality needs to be provided when data resides in storage as well as when data is dynamically accessed by queries. Prior works on query processing on encrypted data did not provide data confidentiality guarantees in both aspects. Tradeoff between secrecy and efficiency needs to be made when satisfying both aspects of data confidentiality while being suitable for practical use. Second, to support common relational data management functions, various types of queries such as exact queries, range queries, data updates, insertion and deletion should be supported. To address these issues, this paper proposes a comprehensive framework for secure and efficient query processing of relational data in the cloud. Our framework ensures data confidentiality using a *salted IDA* encoding scheme and *column-access-via-proxy* query processing primitives, and ensures query efficiency using matrix column accesses and a secure B+-tree index. In addition, our framework provides data availability and integrity. We establish the security of our proposal by a detailed security analysis and demonstrate the query efficiency of our proposal through an experimental evaluation.

Keywords: Data security in the cloud, Query processing on encrypted data, Data confidentiality, Data availability.

1 Introduction

Cloud computing has been gaining interests in the commercial arena due to its desirable features of scalability, elasticity, fault-tolerance, self-management and pay-per-use. However, the security of sensitive data stored in the cloud remains a big concern, and even a road block to the widespread usage of the cloud for relational data management and query processing. The shared environment of the cloud renders access control policies and authentication vulnerable [1]. Many enterprises therefore question whether adequate security and functionality can be ensured for performing their regular data storing and query processing tasks in the cloud.

Data confidentiality is one of the most important security concerns and challenges. It should be adequately provided to safeguard against attackers' analysis and inferences. In addition, data confidentiality has to be balanced with query

© Springer-Verlag Berlin Heidelberg 2011

processing functions and performance. First, although encryption is a commonly used solution to data confidentiality, encryption itself is insufficient to guarantee data confidentiality, even if the encryption scheme does not reveal any characteristics about the plaintext data and is resistant to statistical analysis. When encrypted data is frequently accessed to serve clients' queries, any potential information leakage should also be controlled, since attackers may infer the plaintext data from clients' accessed positions on the encrypted data. Many existing proposals of query processing on encrypted data do not consider both confidentiality for data residing in storage and for data being accessed by queries [2,3,4,5,6,7]. Second, different queries must be supported in the same framework, and practical query performance should not be lost in pursuit of the above data confidentiality requirements. Note that some of the previous works are only able to support one or two types of queries on encrypted data, and in general do not support data updates [3,4]. The powerful cryptographic techniques such as homomorphic encryption [8] and Private Information Retrieval (PIR) [9] can satisfy the above mentioned data confidentiality requirements, but they are computationally expensive and can adversely impact both latency and throughput. The approaches improving the performance of PIR via the use of special hardwares [10,11] may not be feasible for some small businesses who do not have the resources to make such investments.

Next to data confidentiality are the concerns of data availability and integrity. *Information Dispersal Algorithm* (IDA) [12] and similar error-correcting codes [13] have been used in recent works [12,14,15] to provide data availability, and are commercialized [16]. A recent trend in industry even considers IDA as an alternative to traditional data encryption [17], since IDA provides both data availability and a certain degree of data confidentiality.

Our goal in this paper is to provide a comprehensive secure query processing framework that addresses the issues of data confidentiality, availability and integrity, and supports practical processing of various types of queries on relational data in the cloud. We aim at a practical solution with balanced security and functions. We achieve confidentiality for data residing in storage using a variant of IDA, called *"salted" IDA* (Section 4). Salted IDA relies on pseudo-randomness to improve the data confidentiality of the original IDA scheme against computationally bounded adversaries and relies on the original IDA scheme to provide data availability. We achieve confidentiality for data dynamically accessed by queries by transforming query requests to single operations and routing them via trusted proxies, which we call *column-access-via-proxy* (Section 5), so that different queries and queries among different clients are unlikely to be differentiated. We discuss the security implications of these two schemes in a comprehensive security analysis (Section 7).

To enable practical query processing, we build a secure B+-tree index [18] on frequently queried attributes. We encode and disperse the index and the data tuples into matrix column pieces using salted IDA, and access the index and the tuples using the *column-access-via-proxy* operations. During query processing, a client retrieves and decodes only a small part of the index, based on which the client locates the candidate answer tuples. We boost query performance by caching parts of the index on the client. Caching the index also helps improve data confidentiality at accesses by confusing inferences on the index traversal paths. We are thus able to support common relational database queries such as exact queries, range queries and data updates with consistent security guarantees (Section 6) and practical performance (Section 8).

2 Related Work

To support queries on encrypted relational data, one class of solutions proposed processing encrypted data directly. However, these approaches do not provide good tradeoff between data confidentiality and query efficiency. For example, the methods that attach range labels to encrypted data [2,3] reveal the underlying data distributions. Methods relying on order preserving encryption [4,19] reveal the data order. These methods cannot overcome attacks based on statistical analysis on encrypted data. On the other hand, homomorphic encryption is secure and enables calculation on encrypted data [20,8], but relies on expensive public key cryptosystem and thus is not practical.

Instead of processing encrypted data directly, an alternative is to use an encrypted index which allows the client to traverse the index and to locate the data of interest in a small number of rounds of retrieval and decryption [6,7,5]. Although these works provide confidentiality for data residing in storage, they do not provide data confidentiality under dynamic query access patterns. Recent work obfuscates users' data access patterns using special oblivious RAM for data outsourced in the cloud [11], but it still incurs a lot of computation and communication costs and requires special-purpose hardware. A contemporary work to our work obfuscates users' data access pattern by shuffling index nodes [21]. In contrast to the above approaches, our work provides a comprehensive and practical secure query processing framework to protect data in storage and at accesses as well as to support different kinds of queries.

3 System and Attacker Model

3.1 System Model

Data Model. We consider a relational table D with N tuples. Each tuple t has d attributes, $A_1, A_2, ..., A_d$. An index I is built on the frequently queried attributes of D, such as the primary key. Without loss of generality, we refer to I as a one-dimensional index with one-to-one mapping to the tuples in D. We assume each attribute value (and each index key) can be mapped to an integer in the range of $[1, ..., MAX]$.

Data Storage Model. The tuples and the index are encoded under separate secret keys C and then stored on n servers, $S_1, S_2, ..., S_n$, hosted by cloud storage providers. The same keys C are used for decoding the tuples and the index retrieved from servers. The tuples and the index are only accessible to the clients who own the data or the trusted partners of the clients (partners are also referred to as clients hereinafter).

Data Access Model. We assume that the cloud is heavily loaded with many clients issuing many queries continuously. This is typical of modern cloud systems. We support exact, range queries and tuple updates given index keys as predicates, as well as tuple insertion and deletion.

3.2 Attacker Model

Attacker and Prior Knowledge Assumptions. We consider attackers are external entities or the servers where data is stored. We do not deal with insider attacks, such as from malicious partners. We assume client machines are safe, thus any confidential information on the client such as the secret key C is not known to attackers. Attackers do not know clients' queries. However, attackers could know the clients' data distribution and even some exact values and their occurrence frequencies. We assume attackers' computations are bounded by polynomial size circuits.

Attacks. We consider two types of attacks: (1) attacks that target to compromise data confidentiality without compromising data availability or integrity; (2) attacks that target to compromise data integrity or availability, e.g. modifying the encoded tuples or index keys, or Denial-of-Service (DoS) attacks. We say servers are *faulty* in (2). In (1), attackers can compromise any number of servers. They can analyze the encoded data, monitor index and data accesses, and perform inference or linking attacks [6], in which they try to infer the correspondence between the positions of encoded data in storage and plain-text values in the data domain, and even try to infer the secret key C.

4 Data Encryption and Dispersal by "Salted" IDA

Information Dispersal Algorithm (IDA) [12] ensures secure and reliable storage. It is widely used in emerging cloud storages [16,14,22]. We use IDA as the basis for providing data confidentiality and availability, and propose an easy-to-use data encoding and dispersal scheme called *salted IDA*.

4.1 Information Dispersal Algorithm (IDA)

We first introduce IDA [12]. IDA *encodes and disperses data into n uninterpretable pieces so that only m ($m \leq n$) pieces are required to reconstruct the data, and the total storage size of the dispersed pieces is only n/m times of the data size.* Consider that n pieces are distributed onto n servers, then IDA can tolerate up to $(n-m)$ faulty servers (faulty pieces) for data retrievals. Table 1 summarizes the notations we use in the paper.

Table 1. Table of Frequently Used Notations

Notation	Description
n	Number of dispersed data pieces (number of servers to distribute the data)
m	Threshold number of pieces to recover the data (threshold number of servers to retrieve the data)
N	Number of data tuples or keys
d	Number of attributes in one tuple
C	$n \times m$ secret key matrix
ID, TD	Plaintext index matrix, data tuples matrix
IE, TE	Encoded index matrix, data tuples matrix
$E_{i,:}$, $E_{:,i}$	ith row, ith column of matrix E
E^*	$m \times m$ sub matrix obtained by deleting rows in E
b	Number of branches in a B+-tree index node
col	Column address pointing to a column in a matrix
key	Key in a B+-tree index node

Given a matrix M, let $M_{i,:}$ be its ith row, $M_{:,i}$ be its ith column, and $M_{i,j}$ or M_{ij} be the entry at the ith row, jth column of M. Consider an $m \times w$ data matrix D. Each entry in D is an integer in a finite field $GF(2^s)$, or a residue mod $B = 2^s$. The following data values and arithmetic operations are on $GF(2^s)$. To encode and disperse D, IDA uses an $n \times m$ information dispersal matrix C, in which *every m rows are linearly independent, or any submatrix C^* formed by any m rows of C is invertible.* A Vandermonde matrix satisfies this property, where each row is in the form of $C_{i,:} = (1, a_i, ..., a_i^{m-1})$ $(a_i \in GF(2^s), 1 \le i \le n)$. For example in $GF(2^4)$,

$$C = \begin{pmatrix} 3^0 & 3^1 & 3^2 \\ 4^0 & 4^1 & 4^2 \\ 5^0 & 5^1 & 5^2 \\ 6^0 & 6^1 & 6^2 \\ 7^0 & 7^1 & 7^2 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 5 \\ 1 & 4 & 3 \\ 1 & 5 & 2 \\ 1 & 6 & 7 \\ 1 & 7 & 6 \end{pmatrix}$$

Let the encoded data matrix be $E = C \cdot D$, then each row of E, $E_{i,:}$ $(1 \le i \le n)$, is a dispersed piece stored on a server. To reconstruct D, we collect m dispersed pieces, corresponding to m rows of E. Let these rows form an $m \times w$ submatrix of E, E^*. Keep the corresponding m rows of C to form an $m \times m$ submatrix of C, C^*. Then

$$D = C^{*-1} \cdot E^* \tag{1}$$

For example in $GF(2^4)$, consider a matrix $D = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$. Using $C = \begin{pmatrix} 1 & 3 & 5 \\ 1 & 4 & 3 \\ 1 & 5 & 2 \\ 1 & 6 & 7 \\ 1 & 7 & 6 \end{pmatrix}$, we get

$$E = C \cdot D = \begin{pmatrix} 1 & 3 & 5 \\ 1 & 4 & 3 \\ 1 & 5 & 2 \\ 1 & 6 & 7 \\ 1 & 7 & 6 \end{pmatrix} \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} = \begin{pmatrix} 8 & 6 & 7 \\ 12 & 9 & 9 \\ 13 & 10 & 8 \\ 4 & 8 & 8 \\ 5 & 11 & 9 \end{pmatrix}$$

We distribute five rows $E_{1,:}, E_{2,:}, ..., E_{5,:}$ onto five servers $S_1, S_2, ..., S_5$ respectively. If S_2 and S_3 are faulty, we obtain $E_{1,:}$, $E_{4,:}$ and $E_{5,:}$ from S_1, S_4 and S_5 to form E^*. We then delete $C_{2,:}$ and $C_{3,:}$ from C to form C^*, and reconstruct D using Equation (1).

$$D = {C^*}^{-1} \cdot E^* = \begin{pmatrix} 1 & 3 & 5 \\ 1 & 6 & 7 \\ 1 & 7 & 6 \end{pmatrix}^{-1} \begin{pmatrix} 8 & 6 & 7 \\ 4 & 8 & 8 \\ 5 & 11 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$

4.2 "Salted" IDA

IDA ensures data availability, but does not ensure adequate data confidentiality. An encryption scheme with adequate confidentiality should be resistant to statistical analysis on a set of encrypted data. That is, the encrypted data set should not reveal any characteristics of the corresponding plaintext data set.

Based on IDA, we propose a scheme called *salted IDA* to achieve such data confidentiality. As in IDA, a client maintains an $n \times m$ secret matrix C as the information dispersal matrix and the keys for encoding and decoding a data matrix D, where n, m are determined by the client based on the number of servers that she plans to use and the estimated number of non-faulty servers. In addition, the client keeps a secret seed *ss*, and a deterministic function *fs* for producing random factors based on *ss* and the address of data entries on D. We call these random factors *salt*.

Function *fs* feeds *ss* into a pseudorandom number generator (PRNG). Before encoding and dispersing D onto n servers using IDA, for each column of D, $D_{:,i}$, the client calls the PRNG procedure i times, sets the last generated random number as the *salt*, and then adds the salt to each data entry of $D_{:,i}$, $D_{j,i}$ $(1 \leq j \leq m)$. After decoding the encoded data retrieved from m non-faulty servers, the client reconstructs salts by calling *fs* and then deducts these salts from the decoded data entries, recovering D. An alternative to generate salt is to employ a hash function on *ss* and the column index i, $hash(i, ss)$. The security of salted IDA is established in Section 7.1.

5 Secure Cloud Data Access

5.1 Overview

We use salted IDA to encode and disperse the data onto servers in the cloud. To be able to perform queries on salted IDA encoded matrix, we retrieve partial data by retrieving single columns of the matrix as follows.

$$D_{:,i} = {C^*}^{-1} \cdot {E^*}_{:,i} \qquad (2)$$

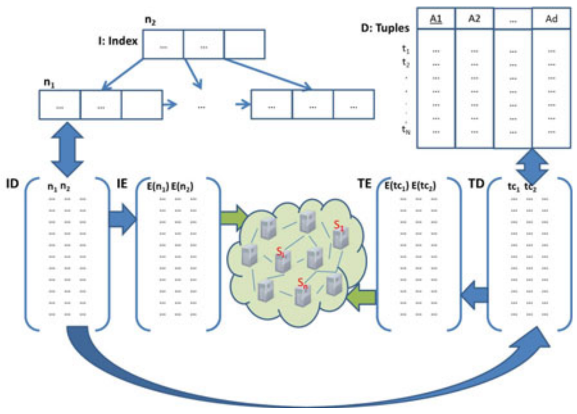


Fig. 1. Secure Cloud Data Access Framework

Similarly we can update and encode a single column $D_{:,i}$ as follows.

$$E_{:,i} = C \cdot D_{:,i} \tag{3}$$

Using the above *column access* property, we can process a query or an update by accessing a few columns at a time. However, selecting which columns to access is still difficult, because searching data directly on the IDA encoded matrix based on plaintext input is infeasible. We solve this problem by building a B+-tree index on the key attribute. The index is kept secure and is only known to the client.

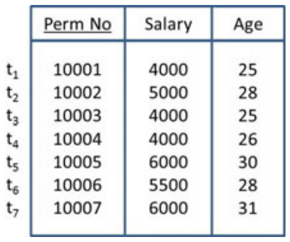
Given a table D with N tuples and a B+-tree index I on the key attributes of D, we store D into a tuple matrix TD, and I into an index matrix ID. TD and ID have a fixed column size, m. Each column of TD thus corresponds to one or more tuples in D. One or more columns of ID correspond to a tree node in I. Each leaf node of I maintains the pointers to the columns of TD where the tuples with the keys in this leaf node are stored. We encode ID into IE and TD into TE, and then disperse IE and TE onto n servers, $S_1, S_2, ..., S_n$, using salted IDA (see Fig. 1). Queries on the index key attribute can be efficiently processed by locating the columns of ID (tree nodes) that store the query keys and then retrieving the corresponding tuples from columns of TD.

5.2 Organization of Index

Let the branching factor of the B+-tree index I be b. Every node of I then has $[\lceil \frac{b-1}{2} \rceil, b-1]$ keys, and every internal node of I has $[\lceil \frac{b}{2} \rceil, b]$ children. We fix the size of a tree node as $2b+1$. Since the column size of the index matrix ID is fixed to m, the ideal case would be $m = 2b+1$, one column for one tree node. We assume the ideal case in this paper and discuss the case of multiple columns representing one tree node in our technical report [23].

We assign each tree node an integer column address denoting its beginning column in ID according to the order it is inserted in ID. Similarly, we assign every tuple column of TD an integer column address according to the order its tuples are added into TD. These column addresses serve as pointers to the tree nodes.

We represent a tree node of I, $node$, or the corresponding consecutive columns in ID, $ID_{:,g}$ as $(isLeaf, col_0, col_1, key_1, col_2, key_2, ..., col_{b-1}, key_{b-1}, col_b)$, where $isLeaf$ indicates if the node is a leaf node. key_i is an index key, or 0 if $node$ has less than i keys. For an internal node, $col_0 = 0$, $col_i\,(1 \leq i \leq b)$ is the beginning column address of the ith child node of $node$ if key_{i-1} exists, otherwise $col_i = 0$. For a leaf node, col_0 and col_b are the beginning column addresses of the predecessor/successor leaf nodes respectively, and $col_i (1 \leq i \leq b-1)$ is the column address of the tuple with key_i.



	Perm No	Salary	Age
t_1	10001	4000	25
t_2	10002	5000	28
t_3	10003	4000	25
t_4	10004	4000	26
t_5	10005	6000	30
t_6	10006	5500	28
t_7	10007	6000	31

Fig. 2. Employee Table

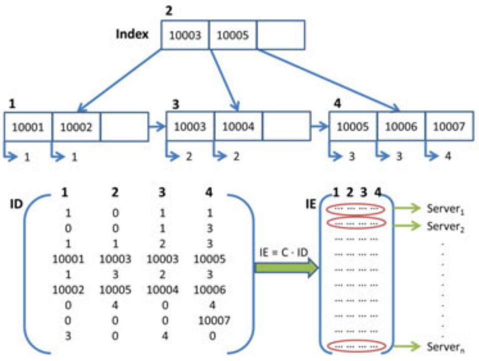


Fig. 3. Index Matrix of Employee Table

Given an example $Employee$ table shown in Fig. 2, Fig. 3 gives an index (the upper part) built on $Perm\ No$ and the corresponding index matrix ID (the lower part). In the figure, the branching factor $b = 4$, and the column size of ID, $m = 9$. Keys are inserted into the tree in ascending order. The numbers shown on top of the tree nodes are the column addresses of these nodes. The numbers pointed to by arrows below the keys of the leaf nodes are the column addresses of the tuples with those keys. For the root node $ID_{:,2}$, $isLeaf = 0$, $col_0 = 0$, $col_1 = 1$ is the column address of its leftmost child, $key_1 = 10003$, $col_2 = 3$ is the column address of its middle child, $key_2 = 10005$, $col_3 = 4$ is the column address of its rightmost child, $key_3 = 0$ and $col_4 = 0$ for no third key. For the leaf node $ID_{:,1}$, $isLeaf = 1$, $col_0 = 0$ for no predecessor, $col_1 = 1$ is the column address of the tuple with key $= 10001$, $col_2 = 1$ is the column address of the tuple with key $= 10002$, $col_3 = 0$ and $key_3 = 0$ for no third key, and $col_4 = 3$ is the column address of the successor $ID_{:,3}$.

5.3 Organization of Data Tuples

To disperse data tuples on the same set of servers as the index keys, the column size of the tuple matrix TD is also set to m. Initially, to organize the existing

d-dimensional tuples of D in TD, we sort these tuples in ascending order of their keys, and pack every p tuples in a column of TD such that $p \cdot d \leq m - k$ and $(p + 1) \cdot d > m - k$, where k is the size of a secure checksum. The checksum is calculated by applying the *Message Authentication Code* (MAC) [24] on the attribute values of all p tuples, so as to verify the integrity of these tuples returned by servers.

After initialization, a new tuple t is inserted in the last column of TD if the column can accommodate t, or inserted into a new column at the end of TD. Tuples are not stored in the order of their index keys as in the initialization. This approach speeds up tuple insertion. A deleted tuple is removed from the corresponding column by leaving the d entries it occupied previously empty (the corresponding encoded entries are not empty, but are filled with salt).

5.4 Secure Column Access via Proxies

In our framework, a client directs query processing, while the servers store or retrieve columns on the index matrix ID and the tuple matrix TD based on the client's decisions. Both data updates and the initial data uploading need to store columns. Read-only queries only need to retrieve columns. To store a column of ID (or TD), $ID_{:,i}$ (or $TD_{:,i}$), the client adds salts into the data entries in the column, encodes the column using Equation (3), and disperses it onto n servers. To retrieve $ID_{:,i}$ (or $TD_{:,i}$), the client retrieves m pieces from m non-faulty servers, decodes the assembled column using Equation (2), and deducts salt. The m requests are sent in parallel.

By monitoring these column accesses, attackers cannot precisely determine the content of a query or the plaintext data involved. However, attackers could learn the initiator client's identity through social engineering attacks, and then infer the client's query and the data accessed in the query. To hide query initiators from attackers, we route column access requests and responses for different clients through a *trusted proxy*, so that attackers cannot even distinguish between different queries sent from different clients. Multiple proxies can be used for load balancing and fault tolerance. A client can switch to another proxy whenever needed. We call this scheme *column-access-via-proxy*. Its security guarantee is analyzed in Section 7.1.

6 Query Processing

Our framework supports exact, range queries, as well as updates, inserts and deletes. These common queries form the basis for general purpose relational data processing.

Exact Query. To find the tuple t for a given index key x, the client traverses the index downwards from the root. This traversal is similar to the traversal on a traditional B+-tree index, except that retrieving each tree node requires retrieving the corresponding index matrix column. At the end of the index traversal, if the client finds x in a leaf node, the client follows the tuple matrix column address associated with x to locate t, which also needs retrieving the column where t is stored.

Range Query. To find the tuples whose keys fall in a given range $[x_l, x_r]$, the client locates all qualified keys in the leaf nodes of the index, gets the addresses of the tuple matrix columns associated with these keys, and then retrieves the answer tuples from these tuple matrix columns. The qualified index keys are located by performing an exact query on either x_l or x_r, and then following the successor or predecessor links at the leaf level. Note that the answer tuples cannot be retrieved directly from the tuple matrix columns in between the tuple matrix columns corresponding to x_l and x_r, since tuples can be dynamically inserted and deleted, and the tuple matrix columns may not be ordered by index keys. After finding the qualified index keys and the associated tuple matrix column addresses, the qualified tuple matrix columns can be retrieved in batch.

Tuple Update. Update to a tuple without changing its index key can be done by performing an exact query on the key to get the target tuple column and then storing the updated tuple column.

Insertion and Deletion. Data insertion is done in two steps: tuple insertion and index key insertion. The corresponding columns in the tuple matrix TD and in the index matrix ID need to be updated by re-storing these columns. Data deletion follows a similar process, with the exception that the tuple to be deleted is first located based on the tuple's key. The order that a TD column is updated before the ID column is important, since the column address of the TD column is the link between the two and needs to be recorded in the ID column. Index key insertion and deletion are always done on the leaf nodes, but node splits or merges may be needed to maintain the B+-tree structure. The overhead in these cases is still small, since the number of nodes (columns) to be updated is bounded by the height of B+-tree, $log_b N$.

Boosting Performance and Improving Data Confidentiality at Accesses by Caching Index Nodes on Client. The above query processing relies heavily on index traversals, which means that the index nodes are frequently retrieved from servers and then decoded on the client, resulting in a lot of communication and computation overhead. Query performance can be improved by caching some of the most frequently accessed index nodes in clear on the client. Top level nodes in the index are more likely to be cached. We assume that the root node of the index is always cached. Caching the index could also confuse attackers' inferences that infer the index structure and the data based on the order of requests, thus help improving data confidentiality during data accesses.

7 Security Analysis

In loaded cloud environment with the use of proxies between clients and servers and client side index caches, we ensure data confidentiality against polynomial size circuits bounded attackers, even when all servers are monitored by attackers. We ensure data integrity and availability when no more than $n - m$ servers are faulty.

7.1 On Data Confidentiality

We rely on the definition of *data indistinguishability* [10,25] to prove the confidentiality of “salted” IDA encoding. *Data indistinguishability* means that *the encryption of any two database tables with the same schema and the same number of tuples should be computationally indistinguishable for any polynomial size circuit.* It is a strong security guarantee in that it invalidates statistical analysis on encoded data. The original IDA scheme [12] does not have such security guarantees, e.g. equal plaintext columns would be encoded into equal ciphertext columns, and constructing $m \times m$ correct correspondences between plaintext and ciphertext data could reveal the secret key C. We show in the following that salted IDA achieves data indistinguishability.

Theorem 1. *If the random numbers generated by a pseudorandom number generator (PRNG) are indistinguishable from truly random numbers, $\forall$ two $m \times w$ data matrices D, D' in $GF(2^{32})$, their encryption under the salted IDA scheme are computationally indistinguishable.*

Proof. Given that the random numbers generated by PRNG are drawn uniformly from $GF(2^{32})$, each column of a matrix $D_{:,i}$ will be added with a salt value which is uniformly distributed in $[1, 2^{32}]$, thus the number of possible choices of salts for each column is 2^{32}. For w columns, the total number of possible choices of salts is 2^{32w}. Since $w > N/(b-1)$, $2^{32w} > 2^{32N/(b-1)}$, which is exponential in N. For a typical database index with $b = 50, N = 10^6$, $2^{32N/(b-1)} > 2^{653061}$. Given such a large choice space of salts and that after adding salt to D and D', the data will be encoded and mixed together by applying an unknown secret matrix C, the ciphertext matrices E, E' obtained by salted IDA encoding are computationally indistinguishable. □

Since the rows of E, E' are distributed onto n servers, two rows $E_{i,:}, E'_{i,:}$ are also computationally indistinguishable. Similarly due to the large choice space of salt, C is unbreakable on polynomial size circuits. However, ensuring the security of the salted IDA encoding scheme itself does not ensure data confidentiality. For example, a target data table may still be located due to its unique size. Then attackers could monitor data accesses on its index matrix and infer the keys based on user accessed positions and known index key distribution. We therefore define confidentiality as follows.

Definition 1. *A* secure relational data processing framework ensures data confidentiality *if it satisfies the following conditions: (1) Two encrypted data sets are computationally indistinguishable; (2) By observing index accesses on the encrypted data, finding out the correct correspondences between the plaintext data and the encrypted data only has negligible advantage over random guesses with prior knowledge on polynomial size circuits.*

Theorem 1 shows that our framework satisfies condition (1). We next show that it satisfies condition (2). Because query processing in our framework is performed through *column-access-via-proxy* operations, and the cloud is typically loaded with many queries from many clients, a client's data access pattern is obfuscated among multiple clients, and a query's data access pattern is obfuscated among multiple queries.

Lemma 1. *In loaded environment with index cache enabled on the client, the best that attackers can gain under the column-access-via-proxy scheme is to identify the columns that represent leaf nodes of the index.*

Proof. In loaded environment with index cache enabled on the client, all the attackers observe on the index are single and batch column accesses. The exact structure of the index are not known to the attackers. To them, single column accesses could correspond to either internal nodes or leaf nodes in processing exact or range queries, while batched column accesses could only correspond to leaf nodes in processing range queries. In the long term, attackers may be able to identify a large number of leaf nodes and sort some of them in the natural order of key values, but they are unlikely to get the total order of all the leaf nodes. □

Based on Lemma 1, we show our framework satisfies condition (2) of Definition 1.

Lemma 2. *Finding the correct correspondences between plaintext keys and encoded leaf nodes only has negligible advantage over random guesses on polynomial size circuits.*

Proof. Assume the exact plaintext key values and the exact order of all the leaf nodes are known. Consider the possible ways of distributing N ordered key values into w ordered leaf nodes with the constraint that each node holds $[\frac{b-1}{2}, b-1]$ keys. After $node_i$ holds $\frac{b-1}{2}$ keys, the next $[\frac{b-1}{2}, b-1]$ keys can only be distributed between $node_i$ and $node_{i+1}$, yielding $\binom{\frac{b-1}{2}+1}{1} = \frac{b+1}{2}$ choices. As there are $(w-1)$ pairs of preceding and succeeding nodes in total, the total number of choices is $(\frac{b+1}{2})^{w-1}$. Since $w > N/(b-1)$, $(\frac{b+1}{2})^{w-1} > (\frac{b+1}{2})^{N/(b-1)-1}$, which is exponential in N. For a typical database index of $b = 50, N = 10^6$, $(\frac{b+1}{2})^{N/(b-1)-1} > 2^{27957}$. Given such a large choice space, finding the correct correspondences between plaintext key values and encoded leaf nodes only has negligible advantage over random guesses. □

Since our framework satisfies Definition 1, we claim the following.

Theorem 2. *The proposed secure relational data processing framework ensures data confidentiality.*

Note that the loaded environment that we assume in the above is typical in the cloud. We do not deal with under-loaded scenarios for now, but we suggest requesting redundant columns in a k-anonymous [26] fashion in each request to provide practical data confidentiality at accesses in under-loaded scenarios.

7.2 On Data Integrity and Availability

We check integrity violations on the index structure using the relationships of sorted key values and the relationships of nodes in the index, and check integrity violations on data values using the checksums. We rely on IDA to provide data availability when no more than $n - m$ servers are faulty. More details on data integrity and availability can be found in our technical report [23].

8 Experimental Evaluation

Our evaluation focuses on the following: (1) the efficiency of our framework for processing different types of queries; (2) the overhead introduced by security when compared with the *baseline* query processing with no security provided, and with the basic encrypted index approach [6] of insufficient data confidentiality and no data availability; (3) the overhead breakdown in terms of client processing time, server processing time and network latency as well as the communication sizes for index and tuples; (4) the effects of data size, query selectivity and index caching on query performance.

8.1 Implementation and Setup

Implementation. We implemented the baseline approach, denoted as *baseline*, the basic encrypted index approach [6], denoted as *encr*, and our approach, denoted as *sida*, in C++. For *baseline*, all the query processing is done on the server and a plaintext B+-tree index is used. For *encr*, a B+-tree index is stored on the server, with each node encrypted using 3DES. We used Crypto++ Library 5.6.0 [27] for implementing IDA and MAC (Message Authentication Code) in *sida* and for implementing 3DES in *encr*. We simulated servers in the cloud by using exactly the same number of local files, and simulated network latency by dividing the communication sizes with the average internet download speed (5.1Mbps) and upload speed (1.1Mbps) in a wide area network [28]. To account for the overhead of proxy in *sida*, we doubled the calculated network latency. We implemented the client side index cache for all three approaches for fairness of performance comparison. Given a client desired cache hit rate, we cached the most frequently accessed index nodes based on the query workload.

Data Set. We extracted 5 attributes, *I_ID*, *I_A_ID*, *I_RELATED1*, *I_STOCK* and *I_PAGE* from the *Item* table of TPC-W Benchmark [29] to form the test data table and built an index on the primary key *I_ID*. We used a TPC-W data generating tool to generate different sizes of tuple sets.

Setup. We fixed the branching factor of B+-tree index $b = 50$. We used $m = 13, n = 21$ servers for *sida*, and only one server for *baseline* and *encr* respectively. Our experimental parameters are summarized in Table 2. For each combination of parameters, we generated 1000 exact queries, range queries, data updates and inserts respectively. A query key was generated by randomly picking a value from the domain of *I_ID* based on Zipf distribution with the specified query skew (default skew=1). For a range query, we used this generated query key as the pivot value, and picked a fixed size query range (query range/selectivity in Table 2) around the pivot value. For an update or insert, the new values of the tuple were generated using the TPC-W tool. The reported results were averaged over 1000 queries of the same type. Experiments were run on Linux servers with Intel 2.40GHz CPU, 3GB memory and Fedora Core 8 OS.

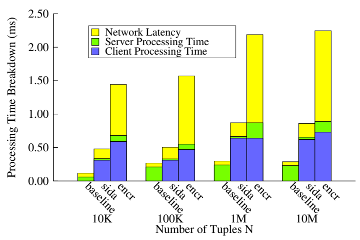
Table 2. Experimental Parameters

Parameter	Domain	Default
Number of Tuples N	$10K, 100K, 1M, 10M$	$1M$
Query Range/Selectivity	100, 500, 1000, 2000	500
Index Cache Hit Rate for Client	0.0, 0.4, 0.8, 1.0	0.8

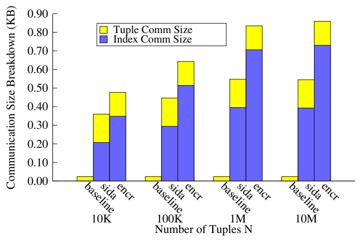
8.2 Experimental Results

General Overhead Comparison. To understand the security overhead due to *sida*, we first evaluate the efficiency of *sida* for processing different types of queries. We varied the number of tuples N from 10K to 10M as shown on the x-axis while fixing other parameters as default. These figures show that having security schemes in *sida* do not dramatically degrade query performance as compared to *baseline* with no security schemes at all. Take 10M tuples as an example, from Figs. 4(a) and 5(a), we can see that the total processing time of *sida* (shown as the middle bar) for an exact query is 0.86ms vs. 0.28 ms of that of *baseline* (shown as the left bar), and the total processing time of *sida* for a range query is 167ms vs. 20ms of that of *baseline*. The communication size of *sida* for an exact query is around 0.5KB vs. 0.023KB of that of *baseline*, as shown in Fig. 4(b), and the communication size of *sida* for a range query is 78KB vs. 9.8KB of that of *baseline*, as shown in Fig. 5(b). In many cases, *sida* even outperforms *encr* which has weaker security guarantees. Although *sida* sometimes transmits more data than *encr* because *sida* packs tuples into tuple matrix columns and uses checksums, as shown in Fig. 5(b), the data communication of *sida* happens between the client and multiple servers in parallel, so *sida* incurs smaller network latency. The comparison result of total processing time on data inserts is similar, so we do not show it here and specifically study its client processing time below. These results suggest that our approach is a practical security solution.

Overhead Breakdown. By breaking down the processing time in Figs. 4(a) and 5(a), we find that client processing dominates query processing in *sida* and *encr*, which is because the client directs query processing and perform all the decoding. For exact queries in which index traversal is the dominant factor, index communication dominates tuple communication, as shown in Fig. 4(b). However for range queries in which tuple processing takes more work than index traversal, tuple communication dominates index communication, as shown in Fig. 5(b).

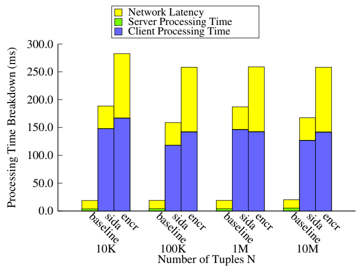


(a) Processing Time Breakdown

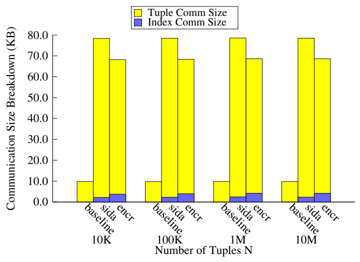


(b) Communication Size Breakdown

Fig. 4. Effects of Varying Number of Tuples N on Exact Queries



(a) Processing Time Breakdown

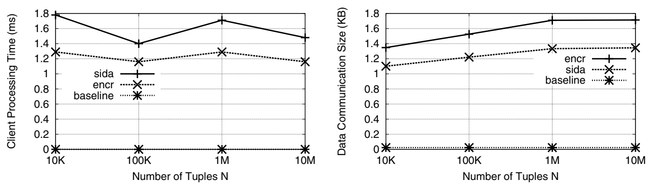


(b) Communication Size Breakdown

Fig. 5. Effects of Varying Number of Tuples N on Range Queries

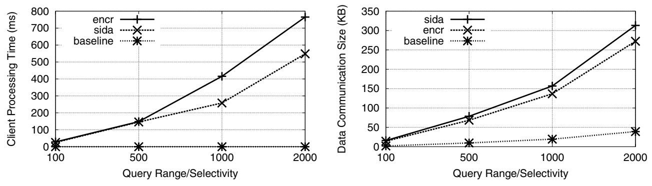
Varying Number of Tuples. We then study the effects of increasing the number of tuples N on query performance. Fig. 4 shows that the processing time and communication sizes for exact queries increase steadily with larger values of N. For data inserts shown in Fig. 6, the client processing time and the data communication sizes increase slowly. The results for data updates are similar to those of data inserts, and thus are omitted due to space limit. However for range queries shown in Fig. 5, these overheads almost do not change. This is because the range query size is fixed, and the major part of processing for a range query is to process the tuples in the requested range, the sum of which could be much larger than the number of traversed index nodes. In general, our approach scales well with the increasing number of tuples.

Varying Query Range/Selectivity. We then study the effects of range query size/query selectivity on query performance. We varied the range query size from 100 to 2000 while fixing other parameters as default. As a result, the answer size for the range query would increase. Fig. 7 shows that the client processing time and data communication size in *sida* and *encr* increase more dramatically than those of *baseline*. This is because *sida* and *encr* must decode the encoded



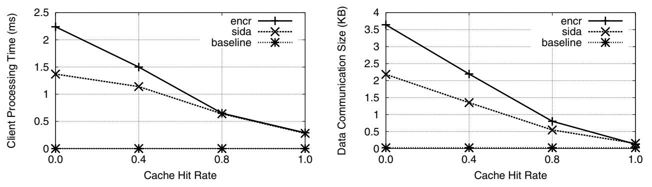
(a) Client Processing Time (b) Data Communication Size

Fig. 6. Effects of Varying Number of Tuples N on Inserts



(a) Client Processing Time (b) Data Communication Size

Fig. 7. Varying Selectivity on Range Queries



(a) Client Processing Time (b) Data Communication Size

Fig. 8. Varying Cache Hit Rate on Exact Queries

candidate answers sent from servers, so they are more sensitive to the change of the query answer size.

Varying Cache Hit Rate. We next study the effects of caching index on the client for reducing the costs for retrieving and decoding index nodes. We changed the desired cache hit rate from 0.0 (no caching) to 1.0 (caching all the index nodes). Fig. 8 indicates that caching improves the performance for processing exact queries.

We have also studied the effects of query skew in reducing the index cache size and the effects of the number of servers on query processing time. We found that the size of index cache needed for 80% cache hit rate is small enough to reside in the client memory, and our framework scales well with the increasing number of servers. These results can be found in our technical report [23].

9 Conclusion

To solve the security concern for widespread use of relational data management in the cloud, this paper has proposed a comprehensive framework for practical secure query processing on relational data in the cloud. Our work is distinguished from previous works in that data confidentiality is ensured in both storage and at access time, and different queries and data updates are supported. Data confidentiality in storage is ensured using the "salted" IDA scheme to encode and disperse the data. Data confidentiality in query accesses is ensured by only allowing proxied single operations called *column-access-via-proxy* between clients and servers. To support efficient query processing, a B+-tree index is built on frequently queried key attributes. Both the index and the data table are organized into matrices, encoded and dispersed using salted IDA. Moreover, data availability is provided using IDA, and data integrity is provided using checksum and the index structure. A security analysis and an experimental evaluation indicate our framework achieves a practical trade-off between security and performance.

Acknowledgement. This work is supported by NSF Grant IIS-1053594.

References

1. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, pp. 199–212. ACM, New York (2009)
2. Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing sql over encrypted data in the database-service-provider model. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, SIGMOD 2002, pp. 216–227. ACM, New York (2002)
3. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004. VLDB Endowment, vol. 30, pp. 720–731 (2004)
4. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. SIGMOD 2004, pp. 563–574. ACM, New York (2004)
5. Ge, T., Zdonik, S.B.: Fast, secure encryption for indexing in a column-oriented dbms. In: ICDE, pp. 676–685 (2007)
6. Damiani, E., Vimercati, S.D.C., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational dbmss. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, pp. 93–102. ACM, New York (2003)
7. Shmueli, E., Waisenberg, R., Elovici, Y., Gudes, E.: Designing secure indexes for encrypted databases. In: IFIP Working Conference on Database Security, pp. 54–68 (2005)
8. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 169–178. ACM, New York (2009)

9. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. Journal of The ACM 45(6), 965–981 (1998)
10. Kantarcioglu, M., Clifton, C.: Security issues in querying encrypted data. In: IFIP Working Conference on Database Security, pp. 325–337 (2005)
11. Williams, P., Sion, R., Carbunar, B.: Building castles out of mud: practical access pattern privacy and correctness on untrusted storage. In: CCS, pp. 139–148 (2008)
12. Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. Journal of The ACM 36(2), 335–348 (1989)
13. Plank, J.S., Ding, Y.: Note: Correction to the 1997 tutorial on reed-solomon coding. Softw., Pract. Exper. 35(2), 189–194 (2005)
14. Bowers, K.D., Juels, A., Oprea, A.: Hail: a high-availability and integrity layer for cloud storage. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS 2009, pp. 187–198. ACM, New York (2009)
15. Wang, C., Wang, Q., Ren, K., Lou, W.: Ensuring data storage security in cloud computing. In: Proceedings of the 17th IEEE International Workshop in Quality of Service, pp. 1–9 (2009)
16. Cleversafe: Cleversafe responds to cloud security challenges with cleversafe 2.0 software release (2010), http://www.cleversafe.com/news-reviews/press-releases/press-release-14
17. www: Information dispersal algorithms: Data-parsing for network security (2010), http://searchnetworking.techtarget.com/Information-dispersal-algorithms-Data-parsing-for-network-security
18. Comer, D.: Ubiquitous b-tree. ACM Comput. Surv. 11(2), 121–137 (1979)
19. Emekci, F., Agrawal, D., Abbadi, A.E., Gulbeden, A.: Privacy preserving query processing using third parties. In: Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, p. 27. IEEE Computer Society, Washington, DC, USA (2006)
20. Ge, T., Zdonik, S.: Answering aggregation queries in a secure system model. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB 2007. VLDB Endowment, pp. 519–530 (2007)
21. De Capitani di Vimercati, S., Foresti, S., Paraboschi, S., Pelosi, G., Samarati, P.: Efficient and private access to outsourced data. In: ICDCS (to appear 2011)
22. Abu-Libdeh, H., Princehouse, L., Weatherspoon, H.: Racs: a case for cloud storage diversity. In: Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC 2010, pp. 229–240. ACM, New York (2010)
23. Wang, S., Agrawal, D., Abbadi, A.E.: A comprehensive framework for secure query processing on relational data in the cloud. Technical report, Department of Computer Science, UCSB (2010)
24. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
25. 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. VLDB Endowment, pp. 127–138 (2006)
26. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report (1998)
27. Dai, W.: Crypto++ library 5.6.0, http://www.cryptopp.com
28. www: report on internet speeds in all 50 states (2009), http://www.speedmatters.org/content/2009report
29. www: Tpc-w, http://www.tpc.org/tpcw

Challenges in Secure Sensor-Cloud Computing

Nayot Poolsappasit, Vimal Kumar, Sanjay Madria, and Sriram Chellappan

Computer Science Department
Missouri University of Science and Technology,
Rolla, Missouri
{nayot,vkq3f,madrias,chellaps}@mst.edu

Abstract. Cloud computing is a concept that has received significant attention lately. With advances in virtualization, coupled with the demand for services, many initiatives are underway in the environment of cloud computing. At this point, most of the services in cloud Computing are broadly in the paradigm of computing and storage, and in the traditional Client Server mode. With the recent explosion of wireless sensor networks and their applicability in civilian and military applications, there is an emerging vision for integrating sensor networks into the cloud. Practical systems like Microsoft's SensorMap and Asia Pacific Environmental Sensor Grid are attestations to the enormous potential for sensor networks to be integrated into the cloud. In this framework users need not own sensor networks. Sensor Network owners after a mission need not disband the networks. There is a symbiotic relationship wherein sensor network owners can provide a variety of services to customers for profit. Customers also benefit from a variety of remote services without being physically close to the environment of interest. However, despite the benefits of sensor-clouds, security issues are largely open. A variety of new threats and attacks are possible, and existing solutions in standalone sensor networks will not be applicable in the cloud. The vision of this article is to introduce the sensor-cloud computing and overview the research challenges from the security perspective.

1 Introduction

Sensor networks are already making impacts to several military and civilian applications, it is natural to believe that end users should also benefit from these services more or less. However, the uses of sensor network is only limited to a small groups of users due to the lack of efficient maintenance cost, elasticity, and the simple deployment (simplicity). With the new computing paradigm of cloud computing adopted in the market. We can then foresee the future demands on sensor based services remotely via the cloud. This demand gives rise to the sensor-cloud computing service which opens the new opportunity of utilizing the on-demand sensing operations and paves ways toward an Internet of things where clients consume a mixture of services from virtual and physical objects.

Sensor-cloud computing is a heterogeneous computing environment in which there are potentially ten of thousands of deployed sensors geographically spread

© Springer-Verlag Berlin Heidelberg 2011

apart. The sensor-cloud provisions the use of physical sensors through the virtualization of sensing services. The virtualization significantly expands the boundary of service capability to facilitate a variety of missions [31,13,18].

Sensor-cloud is particularly attractive as it can change the computation paradigm of wireless sensor networks. As reported in [1], sensor-clouds have various applications including environment monitoring, enterprise computing, scientific simulation, and social networking. For example, hundreds of sensors collaborate toward Microsoft's SensorMap project (http://atom.research.microsoft.com/sensormap), which target a new class of applications forming a large-scale environmental observations by collecting various kinds of data into a central data repository in a continuous, pervasive, and real-time manner. The Asia-Pacific Environmental Sensor Grid (APESG) effort is an initiative that aims to encourage development of technologies for disaster/emergency detection, mitigation, response, and recovery in the Asia-Pacific region, by collecting data from environmental sensors deployed in the participating APEC countries to form a sensor-rich infrastructure.

Sensor-cloud computing surpasses traditional wireless sensor networks in many aspects. In sensor-cloud computing network, users do not need to own sensors. They can simply rent the sensing services. This significantly reduces the cost of ownership enabling the usage large scale sensor networks become affordable. One physical sensor can be projected as multiple virtual sensors and vise versa. The nature of sensor-cloud enables resource sharing and allows virtual sensors scale up or down as needed. Also, the sensor-cloud abstracts different platforms of the physical devices hence giving the impression of a homogeneous network, greatly benefiting users and enhancing satisfaction. Finally, a variety of multi-sensing activities for multiple missions can be simultaneously performed via the cloud greatly enhancing the usability of the devices and networks.

Despite the benefits of the sensor-cloud computing, emerging influences of sensor-cloud could be hindered by various security threats. Hence, the realization of risk in sensor-cloud network will give an early precaution for sensor network application developers to be aware of when joining the sensor network as a service on cloud. The security awareness can lend itself toward the design and integration of security for ensuring trusted computing of the emerging technology. This article identifies newer threats and attacks when sensor networks are integrated with the cloud and propose a spectrum of potential research topics which hold the key solutions to the development of secure sensor-cloud service. The remaining sections are organized as the follow. Section 2 gives an overview of the sensor-cloud service environment. Section 3 identify the new challenges in the delivery of trusted service in sensor-cloud environment from three dimensions namely, secured pre-deployment, secured pre-processing, and secured runtime. For each dimension, we formalize the problem and discuss the spectrum of potential research solutions. Finally, section 5 concludes the approach.

2 An Overview of Sensor-Cloud Service

Sensor-cloud computing is a heterogeneous system combining multiple sensor networks with dedicated purpose for data processing. The sensor-cloud

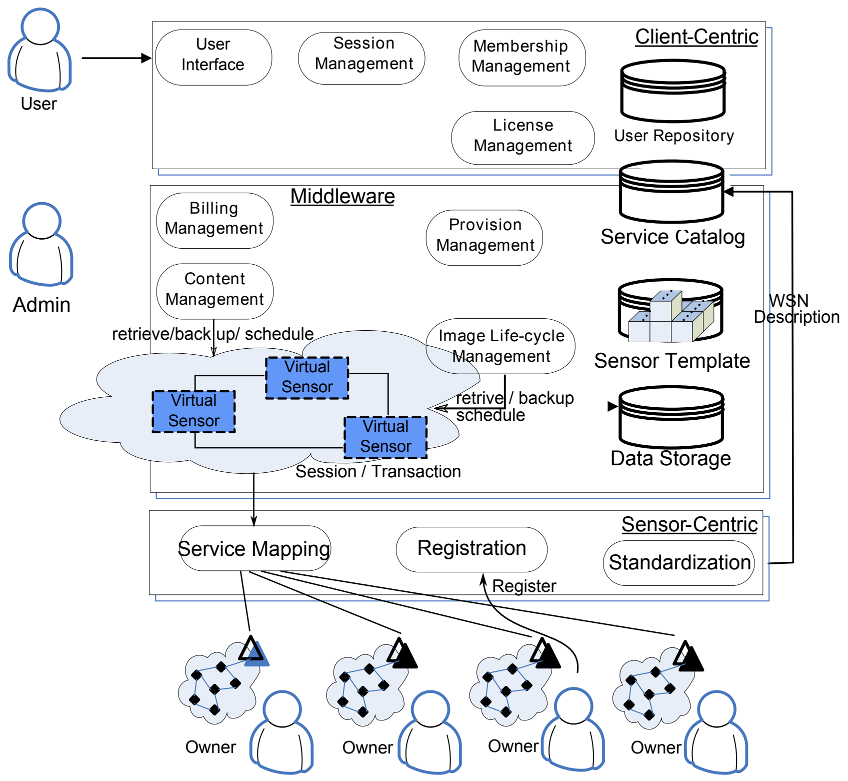


Fig. 1. A Simplified Sensor-Cloud Architecture

provisions the use of physical sensor through the virtualization of sensing services. The virtualization significantly expands the boundary of service capability to facilitate ad hoc missions.

We envision the sensor-cloud network as a heterogeneous computing environment in which there are potentially millions of deployed sensors. A group of sensors is operated by individual organizations/owners. The owners join the cloud and provide sensor-as-a-service. The sensor network may continue sensing a steady stream of information or passively reports the observation when certain conditions are met. In either case, the data are relayed through a gateway and middleware layer network, where sensing data is eventually consumed by clients through the sensor-cloud infrastructure. Figure 1 shows the simplified architecture of the sensor-cloud. The Client-centric layer is responsible for providing service gateway for the users to the working sessions and virtual sensors. The layer provides *user interface* to facilitate users in forming their virtual networks, assembling the workflow, monitoring and collecting data from the virtual sensor network. The middleware administrates service negotiation, virtual sensor network fabrication, service life-cycle, and billing management. Finally, the sensor-centric layer provides the Data-link layer like functions to provision

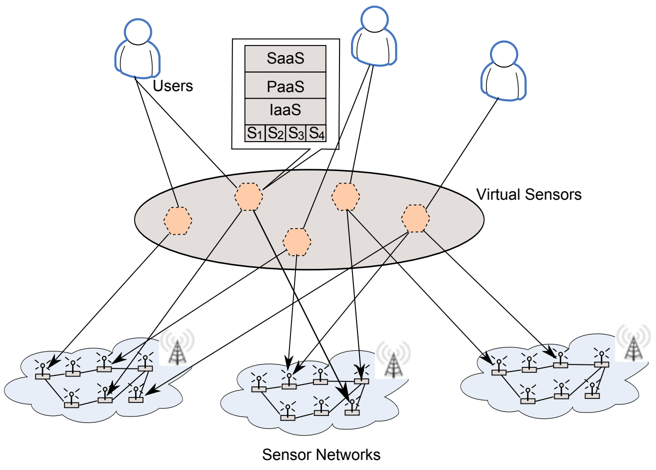


Fig. 2. Virtualization in Sensor-Cloud

the inter connectivity between virtual sensors and the physical sensors. Thus it becomes a gateway communication between virtual sensor and physical sensors (the third party). We envision the virtual sensor as an image of an application which interacts with the users and represent to the user as a custom sensor for a specific mission. The virtual sensors act as a Software as a Service (SaaS) which is responsible for processing the data sensed by the physical sensor. The function of the virtual sensor is to pass the user's specifications to physical sensors and fuse sensing data collected from different sensors and deliver back to the user. In this environment, the user is projected with an illusion that the instance given to him/her is the physical sensor which does the sensing. Below SaaS lies the Platform as a Service (PaaS) providing the computing, networking, and repository environments for SaaS. Finally, Infrastructure as a Service (IaaS) provides the gateway between virtual and physical sensor as depicted in Figure 2. Virtualization can be done in many ways. One-on-one virtualization is when one physical sensor is mapped to one virtual sensor. When multiple physical sensors are mapped to one virtual instance, its referred to as many-to-one. It is also possible for multiple users share the same physical sensors, but through their own instance allocated to them by the middleware. The virtualization projects to the user that they are the sole user of that corresponding physical sensor as well as the abstraction that the user is interacting with the physical sensors directly. The commands issued by the user are passed on to the physical sensors by the middleware. The group which the user creates is visible to him and the administrator who handles the application.

3 Emerging Topics in Securing the Sensor-Cloud Services

Despite the benefits of the sensor-cloud computing, emerging influences of sensor-cloud could be hindered by various security threats. Sensor nodes are susceptible to attacks including node capturing and compromising. Wireless communications can be eavesdropped, captured, or tampered. Most security best-practices cannot be used in WSNs due to the limitation in communication and computation capability. The infrastructure of the sensor-cloud can be misused by malicious users. Malicious user may reveal the physical sensors or physical sensor may reveal the user, which are very undesirable. This implies that the security of sensor-cloud computing is indispensable as the characteristics of this network make it easy for one to abuse, jeopardizing the benefits that could be brought to the society. We now propose emerging research topics in order to ensure security services to the sensor-cloud. In this section, we identify newer threats and attacks from three dimensions: *Secure Pre-Deployment*, *Secure Pre-Processing*, and *Secure Run-time*. For each dimension, we discuss a spectrum of potential solutions integrating concepts from Attack Graphs, Policy Management, Anonymization, Statistical Estimation and Topology aware key management.

3.1 Dimension 1: Secure Preprocessing

The security of the sensor-cloud operations is very important as user interacts with the network/applications directly. Unfortunately, adhering to the security standards does not always guarantee the security on operations which is unique to the sensor-cloud. In this regard, the sensor-cloud allows users to pass the commands to the sensors range from the entire code transfer, script functions, to a seemingly unharmed command argument. If the software does not sufficiently limit which codes or arguments allowing to be passed to a component, it will allow malicious code to be executed leading to potentially security violation. Besides, physical sensor networks can also be seen as countless sources of external risk to the sensor-cloud. In this regard, we need to realize the magnitude in which the cross-layer dependencies influence this kind of risk. In particular, we should study the feasibility of attack to the system from bottom up including how fast and how much does it cost for one compromised state to spread to other.

Related Works. Threat modeling and risk analysis are fundamental to security. Commercial and noncommercial risk analysis tools like SAINT, Nessus, and Snort rely heavily on the list of vulnerabilities. When evaluating security of a network, it is not enough to simply consider the presence or absence of vulnerabilities in isolation. Hence, it cannot correlate local vulnerabilities to depict global vulnerabilities introduced by interconnections between hosts.

In this regard, Attack graph model [30,7,8] has gained more acceptance as it can analyze the threats from both attacker and defender perspectives. This proposal uses attack graph base risk analysis as a key concept for secure pre-deployment.

Emerging Research. Our current research [8,24] have identified that attack graph analysis can help discovering and defining basic system properties that compose system security and other useful attributes; properties that can be verified and validated through theoretical proof and/or experiment. This approach can help system administrator examines the universe of possible consequences following a successful attack.

Using attack graph benefits three folds; the attack graph based risk assessment model reveals the causal dependencies between vulnerabilities and sensor-cloud system properties. Knowing ways in which the sensor-cloud system can be compromised allows the system administrator to be aware of attack surface at the service interface including the risks from external sources. Second, it can help understanding whether the system should be enforced with security policies in what specific component or across components to effectively increase security of the entire system. Third, a given model is formalized as a logical model to realize the relations among the system, security policies, attacks and defensive measures. Finally, the model is rich enough to allow one to measure the risk. Hence, it is an interesting research topic in applying attack graph along with attack surface measurement to formally define metrics for basic system properties and system ability to enforce security policies and defend against known classes of attacks.

3.2 Dimension 2: Secure Processing

Sensor-cloud has objective, asset, and mission to protect. It should have the capability to maintain desired level of security under real world threats. Hence, security administrator needs to ensure that the sensor-cloud has all necessary security controls deployed. In this regards, we have found Identity and Access Management and Information Privacy challenging.

Identity and Access Management. To meet the challenge of distributed sensor-cloud architecture and provisioning toward Business-to-Business collaboration, sensor network owners will eventually extend their local services to external users allowing connectivity to internal users, and sensor-cloud customers. Providing an efficient and seamless data/service outsourcing requires building of trusted service that enables "entities" (include sensor owners and sensor-cloud providers) to securely share their user's identity information and enforces usage restrictions to protect sensitive information. Such service needs to be formed quickly and efficiently to maximize the service productivity and eliminate the need of redundant/successive authentications when the transaction is formed across the organizations. Identity and Access Management refers to such a technology.

Related Works. Several works [20,2,25] have been proposed to address issues in the identity and access management. Among these works, we outline some outstanding products: SAML, Microsoft Forefront Identity Manager, Shibboleth, and OpenID. Security Assertion Markup Language (SAML) [28] is n XML-based standard for exchanging authentication and authorization data between identity provider (IdP) and service providers (SP). Although technology leaders such as Google, Safesforce.com, and Amazon support the SAML protocol only about 5% of the SPs in cloud-computing market appear to support SAML [21]. In Microsoft Forefront Identity Manager 2010 (FIM'10) [29], user's identity is stored in the form of state-based identity, the multi-attributes metadata. These meta-data (claims) act as a security token of a particular user. As opposed to FIM'10, Shibboleth [4] is aiming for decentralized identity management. Each user has a home IdP where SPs can query the profile attributes. The use of Shibboleth is shared among academic institutes. OpenID [25] is a decentralized authentication protocol that helps cloud users managing their multiple digital identities over the sharing of credential information. Given an OpenID, SP queries the IdP from which the OpenID is issued. The major criticism on Shibboleth and OpenID is that they rely on user's judgment on how much information to reveal. Hence, they have been questioned in terms of security and privacy concerns.

Emerging Research. Traditional Federate Identity Services rely heavily on trusted third party. Therefore it is most interested to discover the Identity and Access Management w/o trusted third party. Let's user credential information are described with a set of n attributes $\sum$, Φ_n be the class of predicates used to authorize the request. The problem can be realized as a Hidden-vector encryption [5] corresponding to a predicate encryption scheme for the class of predicates

$$\Phi_n = \{ \phi_{(a_1,\ldots,a_n)} \mid (a_1,\ldots,a_n) \in \textstyle\sum \},$$

where

$$\phi_{(a_1,\ldots,a_n)}(x_1,\ldots,x_n) = \begin{cases} 1\ , if,\ for\ all\ i,\ either\ a_i = x_i\ or\ a_i = don't\ care \\ 0\ , otherwise \end{cases}$$

The theoretical proof of the above equation in regarding to its existence was given in [14]. What was missing from Hidden-vector encryption is the realization on a specific application of identity management. Hence, it is interested for one to focus the research in this direction.

The advantage of this research comes in three folds; first, organizations manage their own policies so that the change of access policy can be done quickly and confidentially. Second, organization can use different criteria implementing the access policy. Beside, the change of the scheme or decision criteria does not require an organization to notify the entire community. Thus, we can provide more effective and secure approach for one to express and restrict access in sensor-cloud computing.

Data Privacy. Data privacy is one of the vital concerns in cloud resource sharing. We know from the past that the advent of data mining technique created a capability for data about individuals to be collected and combined from a wide variety of sources very easily. Therefore the privacy breaches in sensor-cloud service can be difficult to detect as the cloud is provisioned for on-demand services. This implies not only ineffective defenses but also undue cost and wasted resources. In particular, there is a concern that a virtual sensor network can be abused to gather sensitive information or breaching information privacy. This kind of risk is known as *function creep* [26]. The function creep occurs when an item, process, or procedure designed for one specific purpose end up serving another purpose. The function creep in sensor-cloud network is more difficult to prevent.

In addition to the data privacy, we also found that there is a potential for an attacker to breach the meta-data in revealing the identity of wireless sensor network. This problem is similar to Yao's Millionaires' problem in that attackers may take advantage on the service negotiation to reveal the secrets or ownership of the physical sensor networks. In fact, sensors are different in various aspects of device specifications. Taking TelosB and Mica2 sensors for example, we shall see that they are significantly different in memory size, current draw, transmission frequency, and bandwidth. Hence, it is possible for one to craft the sensing operation/mission in such a way that only one specific sensor can operate on it.

Note that we are fully aware that the success chance or severity is highly depended on the level of the prior knowledge of attackers. However, this attack can not be underestimated. The challenge to the problem is how we can balance between the anonymity and the service availability so as to yields the optimal benefit to the sensor-cloud operations.

Related Works. To tackle the privacy attack, data anonymizer techniques like k-anonymity and l-diversity [27,19] appear to be a viable tool. k-anonymity is a property that guarantees to protect the linkage and identification of the data set.

To achieve k-anonymity, several data anonymizer techniques [11,15,3] has been proposed. These approaches base their assumption on single-table data set. Hence, single dimensional k-anonymity approaches are not sufficient for sensor-cloud service as multi relational data set dominates the cloud. In this regard, several multi dimensional k-anonymity techniques are proposed by [9,22,16]. They have shown that Multi relational k-anonymity can be reached although it may not optimal. We are interested to analyze these techniques for the in-transit and at-rest sensor-cloud data set.

One major drawback of k-anonymity is that it does not ensure diversity of sensitive attributes. Hence, it allows adversary to implicitly leak out sensitive information without the need for identifying the identity. Hence, we should consider l-diversity [19] and T-Closeness [17] as an extension of k-anonymity. In particular, sensor-cloud providers should be aware of protentially leaking information through sensitive fields in addition to quasi-identifier fields.

Emerging Research. In sensor-cloud computing, the service catalog and sensor specification are two data sources which are likely to be exploited and, hence, they need to meet k-anonymity property. From k-anonymity's perspective (l-diversity is an extension of k-anonymity on sensitive attributes), the problem is formalized as follow.

Definition 1. MULTIRELATIONAL DATA SET
Let's $\mathcal{C}$ be the service catalog and $\mathcal{S}$ be the standardization table maintaining device specifications of participated sensor networks[1]. *The functional relation $T(A_1, A_2, ..., A_n) = \mathcal{C} \bowtie \mathcal{S}$ be the natural join of $\mathcal{C}$ and $\mathcal{S}$. T describes the universe of system specification and capabilities used by the sensor-cloud in determining if it is capable to provide service as requested by the user. T satisfies*

[1] Service catalog is a public table while the standardization is a private table.

multi relation schema as $\mathcal{S}$ uniquely identifies the existence of the physical sensor network.

Definition 2. K-ANONYMITY OF THE SENSOR-CLOUD DATA
Let's $\mathcal{Q}_{\mathcal{T}}$ be the quasi-identifier of T, representing a set of attributes $\{ A_i, ..., A_j \} \subseteq \{ A_1, A_2, ..., A_n \}$ which could be used for linking with external information to uniquely identify a data entry $t_i \in T$. T is said to satisfy k-anonymity if and only if there are at least k occurrences in the projection of T with respect to $\mathcal{Q}_{\mathcal{T}}$ ($T[\mathcal{Q}_{\mathcal{T}}]$).

Continue along this direction, we need to identify the quasi-identifier and sensitive attributes. In addition, we also need to find out the attributes domain of the requirement properties used to describe the sensing operation/mission in the service negotiation as these attributes can be used as a tool for an attacker to breach the data privacy. Next, we need to analyze various data anonymity techniques so as to discover applicable data prevention approach to be applied to the data repository. These summarize the research challenges in this direction.

3.3 Dimension 3: Secure Runtime

To secure the runtime service, we envisage the importance of the research regarding to, at the minimum, trusted data aggregation and key management.

Trusted Data Aggregation. Once a wireless sensor is physically compromised by an adversary, trust and reputation based data aggregation techniques have been proposed to address it. Most trust models assume firm network connectivity, where each sensor monitors the behavior of its neighbors. Such an assumption is prohibitive in a sensor-cloud environment. Sensor-cloud is a network of networks, and sensors provisioned for a specific application may be spread across a number of networks. This topology presents three challenges in trusted data aggregation: a) individual sensor network owner may not allow sensors from a different network to monitor its sensors. b) sensors may be located far away from each other and may have no means of directly monitoring their virtual neighbors. c) an individual sensor network may be compromised on the whole, which will render the "monitor your neighbor" approach useless.

Related Works. Only a few researched have addressed the problem of trusted data aggregation on virtual network environment. An agent based trust model is discussed by Boukerche et. al [6], where a mobile agent is used to calculate trust on every entity. The mobile agent is supposed to be tamper proof, secure and trusted. However, it is difficult to guarantee these assumptions amidst compromised nodes. Probst et. al [23] propose an approach where nodes evaluate other nodes by comparing their own data with others data. Trust is established based upon how closely the data matches. The scheme though is only designed for faulty sensors and does not take into account, compromised sensor. In [32] a probability distribution function of a group of nodes is generated by the aggregator and sensors data is compared against this distribution statistically. Sensors which follow the distribution are tagged trustworthy while others who dont are not. While this approach works well when a single node is compromised, an adversary which compromises many nodes together can circumvent this scheme.

Emerging Research. From our perspective, remote monitoring and the trusted core are on top of the interested topics. In remote monitoring, sensors are clustered into correlated groups base upon the correlation of the sensed data, the geo-spatial of data entities. From the view of the trust observer, groups are formed again and again at every end of the iteration. A node is trustworthy if it remains in the same group after iteration n as it was after iteration $n-1$. A node changing his group signals deviation from normal behavior. In regarding to remote monitoring approach, we can outline two important challenges, namely, the Correlation estimation and Trusted core.

Correlation Estimation. A research problem is how to find out the correlation between data coming from various sensors. The correlation estimation can be formalized as follow. The aggregator collects data from its children and maintains a vector of past data of each of its child. Let the data sensed by a node i at time t_j is denoted by x^i_j. The aggregator keeps a vector $v_i[x^i_1, x^i_2, \ldots, x^i_n]$ for each of its k children, where n is the size of the window for which the data is kept. A vector $V[v_1, v_2, \ldots, v_k]$ of such vectors is created and a covariance matrix of this vector is calculated. The element at i, j position of this matrix gives us the correlation between v_i and v_j. Based on their correlation, the sensors can now be divided into groups using any clustering algorithm.

Trusted Core. For applications where the setup time x is very large, a long history of each sensor would be required to build correlation. A trusted core of sensors can be used to ease the computation overhead cost. We can make use of the centralized characteristic of the sensor-cloud to establish nodes in each network which act like a trusted core. The trusted core is made up of a handful of well connected nodes. It randomly samples the sensors by sending a query, to which all the other sensors reply with their measurements, irrespective of whether they are working for the application or not. The trusted core then evaluates the trustworthiness of nodes working for the application and sends the report to the aggregators. However, the challenge to trusted core is the optimal node placement. One of the required conditions is that a trusted core node should have a very high connectivity and it should be directly connected to a number of nodes in the network it resides. Another condition is that a trusted core node should be able to connect to other trusted core nodes in other networks. Thus choosing a trusted core node becomes a multi criteria optimization problem. The conditions can be mapped to optimization criteria $\mu_1(x)$, $\mu_2(x)$, ..., $\mu_n(x)$, and an objective can be established as $\min[\mu_1(x), \mu_2(x), \ldots, \mu_n(x)]$. This multi-criterion optimization is an emerging research waiting to be formalized and explored.

Key Management. Confidentiality in sensor network missions is important. In the sensor-cloud service environment, confidentiality manifests in newer dimensions. Users of any sensor-cloud will desire confidentiality of their tasks and communications, which means a desired level of the security needs to be provided for users. Since multiple applications may run simultaneously on a sensor network at the same time, information exchanged among sensors from one application must be isolated from another application. Also, networks of sensor networks may have to collaborate for the same mission. An adversary compromising one sensor network should not be able to compromise collaborating sensor networks. All these situations are unique to sensor-cloud based infrastructures and need new solutions.

Related Works. In a majority of sensor network deployments, sensor positions cannot be controlled especially for large scale missions. Even if there is some control over the placement of sensors, over time, faults and failures can affect sensors. Consequently, a lot of investment has gone in the study of secure key management for randomly deployed sensor networks. Currently, the well accepted approach for key management in WSNs is based on the idea of key pre-distribution (generalized as *KP* protocols) [10]. In the simplest version, each sensor is pre-distributed with k distinct keys randomly chosen from a large pool of K keys. Post deployment, neighboring nodes use the pre-distributed keys to establish a pairwise key in between either directly or using other nodes as proxies. The basic redundancy in initial key pre-distribution enables nodes to overcome deployment randomness, making it easier to discover secure neighbors and proxies. A host of key management protocol variants have been proposed based on key pre-distribution [10] etc., each one improving upon one or more features like *connectivity*, *resilience*, *overhead* etc. Denoting RC (*Resilient Connectivity*), as the Probability that a secure pairwise key is present between two physically neighboring nodes, we have

Theorem 1. *For any KP protocol, and a non-zero node capture probability* $(P_c > 0)$, (1) $\exists Densities\ D_1, D_2 : D_1, D_2 \in (0, +\infty), D_1 > D_2 : RC(D_1) < RC(D_2)$; (2) $lim_{D \to +\infty} RC(D) = 0$.

The first part of Theorem 1 states that for any non-zero node capture probability P_c, performance of KP protocols does not always increase with node density D. There exists densities D_1 and D_2, where RC at a smaller node density is higher than RC at larger node density for any protocol and network parameters. The second part of the theorem states that $RC \to 0$ when $D \to \infty$. It implies there is a finite value of node density D for optimal performance for any KP protocol. To conclude, KP protocols are not scalable with respect to node density in terms of security.

Emerging Research. We found the issue of secure key management in sensor-cloud environment most interested. It is our conjecture that Key Distribution schemes despite inherent limitations are still relevant in the sensor-cloud. Missions in a sensor cloud environment are dynamic, and consequently, pre-deployment of static keys by the base-station of the sensor network is not possible. Furthermore, users may also have changing requirements, which necessitates dynamic key assignments. Since sensor networks are mission oriented, the scalability problem can be alleviated by appropriate assignment of keys. Furthermore, a critical challenge in the cloud environment is collaboration among multiple sensor networks. Therefore, one may direct the research on the investigation of differentiated key predistribution. The idea is to provide different number of keys to different sensors, via different key pools with slight overlap. For two sensor networks S_1 and S_2, let's the K_1 be the key pool for sensor network S_1 and K_2 be the key pool for sensor network S_2. The intersection of K_1 and K_2, denoted as K_3 is small, but not null. Isolating attacks across collaborating networks, while still ensuring secure communications among networks is a very challenging topic.

Continue along this direction, one might interested to study a 2 layer communication framework. Nodes within a network reside at level one. They use key K_1 to communicate amongst themselves. When nodes send information across networks, they send messages to trusted nodes with keys from their respective key pools. The trusted nodes then use keys from key pool K_3 to communicate across networks. Assuming trusted nodes are secure, attacks should be minimal in this framework. Alternatively, the idea of differentiated key distribution can be applied even on a single WSN. This provides us with links with different security and resilience with that network. These links can be chosen for routing. This approach is a cross-layer approach wherein the key management scheme can be combined with the routing layer. This, however, increases the communication overhead of some nodes in the networks. This further increases energy consumption among those nodes. This is an open issue to be investigated.

4 Summary of Research Directions

Sensor-cloud computing emerges with a unique opportunity for abuses and attacks. This implies that the security of sensor-cloud computing is indispensible as the characteristics of this network make it easy for one to abuse, jeopardizing the benefits that could be brought to the society. The broad motivation of this article is an investigation of the security issues and challenges in the sensor cloud environment, and to discuss potential research topics to secure emerging sensor-clouds against a variety of attacks. The core research spread across three directions as described below.

– *Secure Pre-deployment:* This direction involves research on security issues prior to deployment of sensor-cloud. Sensor-cloud developers need to realize the relations among the system, security policies, attacks and defensive measures. Regarding to this, attack graph analysis can be used to examine the universe of possible consequences following a successful attack and determine whether the system should be reinforced, and if so in what specific component (or across components) to effectively increase security of the entire system.
– *Secure Pre-processing:* This direction involves research on security issues prior to execution of missions. To maintain the desired level of security, we need to ensure that the sensor-cloud has all necessary precautions deployed in it before execution. In this direction, we found the predicate encryption

scheme very attractive in dealing with authentication and authorization and also found k-anonymity on multirelational data set an emerging research.
- *Secure Runtime:* This direction involves study of issues related to sensor-cloud security at run time. In particular, we have found demands on secure data aggregation and 2 layer key management framework.

Acknowledgment. The authors would like to thank Mukund Krishnan for his great contribution to the research project.

5 Conclusion

This article introduces the sensor-cloud service and identifies newer threats and attacks from three dimensions: *Secure Pre-Deployment* to proactively identify design problems and reinforce policies and define defense mechanisms; *Secure Pre-Processing* to ensure all safety procedures are in place, along with Identity, Access Management and Information Privacy at mission time; *Secure Run-time* to ensure trusted sensor network operation and key management techniques for cloud based sensor networks. For each dimension, we design a spectrum of potential solutions integrating concepts from Attack Graphs, Policy Management, Anonymization, Statistical Estimation and Topology aware key management.

References

1. Beng, L.H.: Sensor cloud:towards sensor-enabled cloud services, intelligent systems center, nanyang technological university (April 13, 2009)
2. Backhouse, J., Halperin, R.: Approaching interoperability for identity management systems. The Future of Identity in the Information Society, 245–268 (2009)
3. Bayardo, R., Agrawal, R.: Data privacy through optimal k-anonymization (2005)
4. Bhargav-Spantzel, A., Squicciarini, A., Bertino, E.: Trust negotiation in identity management. IEEE Security & Privacy 5(2), 55–63 (2007)
5. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
6. Boukerche, A., Li, X.: An agent-based trust and reputation management scheme for wireless sensor networks. In: IEEE Global Telecommunications Conference, GLOBECOM 2005, vol. 3, p. 5. IEEE, Los Alamitos (2006)
7. Dantu, R., Loper, K., Kolan, P.: Risk management using behavior based attack graphs. In: Proceedings of International Conference on Information Technology: Coding and Computing, ITCC 2004, vol. 1, pp. 445–449. IEEE, Los Alamitos (2005)
8. Dewri, R., Poolsappasit, N., Ray, I., Whitley, D.: Optimal security hardening using multi-objective optimization on attack tree models of networks. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 204–213. ACM, New York (2007)
9. Domingo-Ferrer, J., Torra, V.: Ordinal, continuous and heterogeneous k-anonymity through microaggregation. Data Mining and Knowledge Discovery 11(2), 195–212 (2005)

10. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS (November 2002)
11. Fung, B., Wang, K., Yu, P.: Top-down specialization for information and privacy preservation. In: 21st International Conference on Data Engineering, ICDE 2005, pp. 205–216. IEEE, Los Alamitos (2005)
12. Gu, W., Bai, X., Chellappan, S.: Scaling laws of key pre-distribution protocols in wireless sensor networks. Technical report, Technical Report, The Department of Computer Science, Missouri University of Science and Technology (2010), http://web.mst.edu/chellaps/papers/gu_scaling_techrep10.pdf
13. Kapadia, A., Myers, S., Wang, X., Fox, G.: Secure cloud computing with brokered trusted sensor networks. In: International Symposium on Collaborative Technologies and Systems (CTS) 2010, pp. 581–592. IEEE, Los Alamitos (2010)
14. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
15. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, pp. 49–60. ACM, New York (2005)
16. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, p. 25. IEEE, Los Alamitos (2006)
17. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: IEEE 23rd International Conference on Data Engineering, ICDE 2007, pp. 106–115. IEEE, Los Alamitos (2007)
18. Lim, H., Iqbal, M., Ng, T.: A virtualization framework for heterogeneous sensor network platforms. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, pp. 319–320. ACM, New York (2009)
19. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) 1(1), 3 (2007)
20. Mahalle, P., Babar, S., Prasad, N., Prasad, R.: Identity Management Framework towards Internet of Things (IoT): Roadmap and Key Challenges. Recent Trends in Network Security and Applications, 430–439 (2010)
21. Messmer, E.: Cloud-based identity management gets a boost Network World (May 19, 2010)
22. Nergiz, M., Clifton, C., Nergiz, A.: Multirelational k-anonymity. IEEE Transactions on Knowledge and Data Engineering 21(8), 1104–1117 (2009)
23. Probst, M., Kasera, S.: Statistical trust establishment in wireless sensor networks. In: International Conference on Parallel and Distributed Systems 2007, vol. 2, pp. 1–8. IEEE, Los Alamitos (2009)
24. Ray, I., Poolsappasit, N., Dewri, R.: An Opinion Model for Evaluating Malicious Activities in Pervasive Computing Systems. In: Atluri, V. (ed.) DAS 2008. LNCS, vol. 5094, pp. 297–312. Springer, Heidelberg (2008)
25. Recordon, D., Reed, D.: OpenID 2.0: a platform for user-centric identity management. In: Proceedings of the Second ACM Workshop on Digital Identity Management, pp. 11–16. ACM, New York (2006)
26. Sorniotti, A., Gomez, L., Wrona, K., Odorico, L.: Secure and Trusted in-network Data Processing in Wireless Sensor Networks: a Survey. Journal of Information Assurance and Security 2(3), 189–199 (2007)

27. Sweeney, L.: Achieving k-anonymity Privacy Protection using Generalization and Suppression. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(5), 571–588 (2002)
28. S. URIs. Security Assertion Markup Language (SAML) V2. 0 Technical Overview (2008)
29. Vilcinskas, M., Craw, L., Brekkan, B.: Understanding Forefront Identity Manager 2010 Microsoft Corporation (October 2009)
30. Wang, L., Noel, S., Jajodia, S.: Minimum-cost network hardening using attack graphs. Computer Communications 29(18), 3812–3824 (2006)
31. Yuriyama, M., Kushida, T.: Sensor-cloud infrastructure-physical sensor management with virtualized sensors on cloud computing (2010)
32. Zhang, W., Das, S., Liu, Y.: A trust based framework for secure data aggregation in wireless sensor networks. In: 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, SECON 2006, vol. 1, pp. 60–69. IEEE, Los Alamitos (2007)

SensorSafe: A Framework for Privacy-Preserving Management of Personal Sensory Information

Haksoo Choi, Supriyo Chakraborty,
Zainul M. Charbiwala, and Mani B. Srivastava

University of California, Los Angeles
{haksoo,supriyo,zainul,mbs}@ucla.edu

Abstract. The widespread use of smartphones and body-worn sensors has made continuous and unobtrusive collection of personal data feasible. This has led to the emergence of useful applications in diverse areas such as medical behavioral studies, personal health-care and participatory sensing. However, the nature of highly personal information shared with these applications, together with the additional inferences that could be possibly drawn using the same data leads to a variety of privacy concerns. This paper proposes SensorSafe, an architecture for managing personal sensory information in a privacy-preserving way. Our architecture consists of multiple remote data stores and a broker so users can retain the ownership of their data and management of multiple users can be well supported. SensorSafe also provides a context-aware ne-grained access control mechanism by which users can dene their own sharing rules based on various conditions including context and behavioral status. We discuss our design of the SensorSafe architecture and provide application examples to show how our system can support user privacy.

Keywords: Information Privacy, Personal Sensory Information, Data Management Architecture.

1 Introduction

Mobile smartphones and body-worn sensors have enabled the continuous collection of sensory information about individuals as they live their daily lives. Current smartphones are typically equipped with GPS, WiFi, and accelerometer which can provide location and activity information. Wearable sensors such as BioHarness BT [7] include ECG, respiration, and skin temperature sensors. A variety of inferences can be made by applying machine learning algorithms on the collected data. For example, stress and smoking behaviors can be detected from ECG and respiration data [31], current transportation mode can be determined by using GPS and an accelerometer [33], and personal exposure to pollutants can be measured by using location data together with a public pollutant map [28]. Collection of such sensor data and inferences have other useful applications in areas such as medical behavioral studies [31,1], personal health-care [6,22], and location sharing social applications [36].

© Springer-Verlag Berlin Heidelberg 2011

An important aspect of such applications is that they involve *sharing* of personal sensitive information, which raises significant concerns about an individual's privacy. In behavioral studies, participants share their data with researchers or doctors. Location sharing applications involve friends or family members. Personal health-care applications have coaches who give useful advice about the user's health [6]. Medical home-care systems can involve not only the patient's doctor but also the insurance company [22]. Sharing of personal data is inevitable because it is essential for the application to work. While users want to share some of their personal information to benefit from certain services, they do not want to share information that they feel uncomfortable sharing.

Especially in medical behavioral studies involving multiple institutions [1], Institutional Review Board (IRB) requires that data collected from human participants should be hosted by the institution conducting the data collection [3]. Therefore, it is not possible to have a single centralized server to store data from multiple institutions. Instead, each institutional server should store its own data and interact with other institutional servers to share the data. A data management framework should be able to support such kind of IRB regulations.

A recent user study on awareness about privacy implications of sensory information [32] reports how users' privacy concerns change with personal stake and abstraction levels. In the study, users live their daily lives while wearable sensors and smartphones collect information about exercise, places, conversation, commuting, and stress. In general, the better users understand their data and the personal stakes associated with it, the higher are the concerns regarding its sharing. Especially, certain types of information such as conversation, commuting, and stress lead to more concerns than exercise and places. Privacy concerns also increase as the data contains more specific information such as place, duration, and timestamp. The fact that users have different levels of concerns depending on the types of information shared and the levels of abstractions motivates the need for a privacy-preserving data sharing framework.

Sharing sensory information poses a new challenge in protecting an individual's privacy. Traditional privacy research has focused on preventing de-anonymization of published personal data. This research has been motivated by several incidents such as de-anonymization attacks to Netflix data [29] and the AOL search records [19]. Several privacy metrics [35,25,24] have been proposed, and mechanisms to achieve certain privacy requirements have been devised [16]. While the traditional research protects an individual's identity, sharing of sensory information requires protection of an individual's *behavior*. That is, users want to have complete control over what kind of behavioral information is shared when they provide their sensor data. In addition, identity is often essential information in applications such as behavioral studies or health-care systems. Therefore, protecting private behaviors in sensory information becomes an important issue.

In this paper, we propose SensorSafe, an architecture that enables sharing of personal sensory information in a privacy-preserving way. Our architecture provides users with the ability to control the amount of behavioral information they want to share. This control is achieved by a context-aware, fine-grained access control mechanism which provides numerous options to support various privacy preferences of users. Moreover, the SensorSafe architecture stores sensor data in multiple distributed servers such that users or institutions can have the ownership of their own data. Because data are not stored in a centralized server, managing data from number of users is a non-trivial problem. SensorSafe supports multiple users with ditributed data storage by having a separate broker server.

The rest of this paper is structured as follows. Related work is discussed in Section 2 and important design considerations are presented in Section 3. In Sections 4 and 5, we provide an overview and details of our architecture. We discuss application examples in Section 6 and conclude in Section 7.

2 Related Work

Several privacy breaches have been published [29,19,30], and these incidents have lead to research in protecting privacy of users when their information is shared. In an effort to protect an individual's identity in context of relational data, several privacy metrics such as k-anonymity [35], l-diversity [25], and t-closeness [24] have been proposed. Algorithms to achieve certain privacy requirements defined by the privacy metrics are also proposed. They include perturbation, suppression, generalization, and so on [16]. These research efforts mainly deal with protection against de-anonymization attacks on personal data sets. Although the data sets do not contain explicit identifiers (e.g., name, social security number), the de-anonymization attacks exploit quasi-identifiers (e.g., zip code, age, gender) which cannot be removed due to utility of the data sets. While these privacy metrics and algorithms are useful in context of relational data, they cannot be directly applied to sensory information due to several reasons. First, sensor data are often both sensitive and quasi-identifying so it is harder to anonymize without degrading much of its utility. Second, sensory information often need to be shared with identity (e.g., health-care application, medical behavioral studies) [32].

In order to protect identity when sharing sensory information, many techniques have been proposed. Several works try to preserve aggregated information by modifying original sensor data. AnonySense [14] has a *mix-network* in their architecture which anonymizes sensor data from multiple users. PoolView [17] is an architecture with perturbation scheme which adds noise to original sensor data but maintains a community average and distribution. Later, Ahmadi et al. [8] proposed a data transformation scheme which preserved a regression model of the original data. There are also several techniques for protecting identity that can be inferred from location information. Hoh et al. [20] achieve k-anonymous location updates using a temporal cloaking scheme. Krumm et al. [23] introduced techniques such as deleting, rounding, and addition of noise for obfuscating home location. Although these works protect privacy of identity, they do not deal with privacy of behavioral information in sensor data.

In online social networks, several architectures provide users with control over their own data. Caceres et al. [11] proposed *Virtual Individual Servers* that allows users to retain ownership of their data and to determine what data is shared with whom. PrPl [34] is a decentralized social networking infrastructure with personal data storage called *Personal Cloud Butler*. Lockr [37] provides an access control mechanism based on digitally signed social relationships. Persona [9] provides an access control mechanism via attribute-based encryption with out-of-the-band key exchange. These architectures provide access control that determines who has access to what, but more fine-grained way of access control is needed when sharing sensory information. Fine-grained access control is proposed in Locaccino [36]. However, they lack access control based on a user's context or behavior which is important to protect privacy of behavioral information in sensor data. Commercial software such as Microsoft Health Vault [5] or Google Health [2] also provide sharing of personal data with privacy in mind. However, they are designed for sharing Personal Health Records [38] rather than sensory information.

Mun et al. proposed Personal Data Vault (PDV) [27], which is an individual data storage with fine-grained access control mechanism, privacy rule recommender, and trace audit. Our system enhances the fine-grained access control by supporting privacy rules with context/behavior conditions and control for levels of inferences. In addition, while PDV is a single personal data storage, our architecture facilitates management of multiple individual data stores by having a broker server.

3 Design Considerations

In this section, we discuss several important considerations that have guided us to design SensorSafe. The key functionalities of SensorSafe are storing personal sensory information of *data contributors* who are willing to provide their data and sharing those data with *data consumers* who are interested in such information. The following design considerations are essential for our system to be privacy-preserving, practically usable, and effective.

Selective Sharing: Even though data contributors are willing to share data, they do not want to share all their data because some of the collected information might disclose private and sensitive aspects of their lives. Therefore, our system provides a mechanism for data contributors to share only what they want to share. The key challenge is how a selective sharing mechanism can fully support a data contributor's various privacy preferences. SensorSafe achieves this goal by providing numerous options such as context/behavior, consumer identity, location, time, and levels of inferences.

Data Ownership: Traditional data collection systems store the data in a centralized server [13,28,26]. However, the central server is the single point of failure because if the server is compromised, all the data contributors' information is at risk. Moreover, the IRB regulation mentioned in Section 1 requires multiple institutional servers. The SensorSafe architecture is distributed so the actual storage points can be data contributors' personal computers or institutional servers.

Managing Multiple Data Contributors: Each contributor's data is stored at a different physical server, making the task of data management across contributors a challenging one. Without architectural support, data consumers will have to directly contact each data contributor's server and manually manage them. Especially in behavioral studies, researchers need to search for data contributors with suitable privacy preferences because some contributors might not share enough data for the study. In SensorSafe, we have a dedicated server for managing multiple data contributors and searching for data contributors with suitable privacy rules.

User Controllable Privacy: Studies have shown that users often have difficulty expressing and managing their own privacy policies [15]. Moreover, as data collection at the contributors occurs as they live their daily lives, the privacy preferences tend to change over time. Thus, it is necessary to provide a user-friendly interface that allows the data contributors to control their information at all time. To achieve this, we have designed a web-based user interface where the users can define and manage privacy rules. The user interface consists of standard HTML UI components and Google Maps, which most Internet users are already familiar with.

Data-Store Functionality: It is important for data-storage systems to be general enough to support a variety of applications. Therefore, it should provide enough functionality to support applications. First, a data retrieval mechanism should not limit kinds of queries that applications can issue. Second, data storage should be able to store various types of data that applications require. To achieve this, SensorSafe provides expressive data query language and does not have restrictions on the structure of data.

4 Architecture Overview

The overall system architecture is presented in Figure 1. SensorSafe consists of remote data stores and a broker, which interact with data consumers and data contributors. Data contributors carry smartphones and possibly wearable sensors on their body. The smartphones upload collected sensor data to the data contributors' remote data stores. The data contributors create privacy rules on their data store. When the data contributors are first registered on their data store, they are automatically registered on the broker, too. Data consumers interact with the broker to search for data contributors with suitable privacy rules. After obtaining a list of the data contributors, data consumers directly communicate with remote data stores to download pertinent data. In this process, access to the data is controlled by the data contributors' privacy rules. The broker is not a performance bottleneck because sensor data are directly transferred from each remote data store to data consumers.

Figure 2 illustrates the design of remote data stores and the broker. Every interaction with both servers has to go through the user authentication layer to

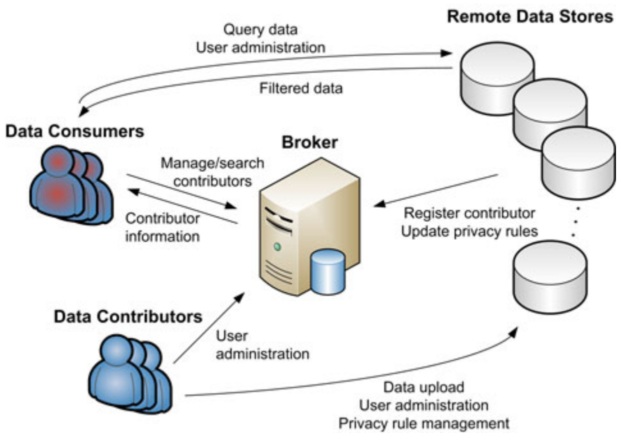


Fig. 1. SensorSafe Architecture

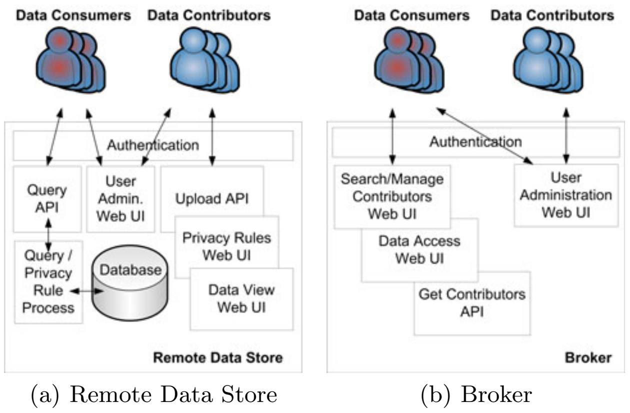


Fig. 2. Remote Data Store and Broker

limit access to registered users only. Both servers also have a web user interface for user administrations. Data consumers access a contributor's data through query API provided by remote data stores. Every access is regulated by the query/privacy processing module, which interacts with the underlying database. Data contributors upload their sensor data through upload API, create/manage their privacy preferences, and view their own data using the web-based interface. Data consumers use the web interface provided by the broker to search and manage data contributors. They can also access a contributor's data through the web user interface. A list of data contributors and their data store locations can be obtained by API on the broker. The architectural details of SensorSafe are discussed in the following sections.

5 SensorSafe Framework

The main functionalities of SensorSafe include remote data stores, the broker, a context-aware fine-grained access control, data contributor management/ searching, and privacy rule-aware data collection. Each component of SensorSafe is discussed in detail as follows.

5.1 Remote Data Stores

Traditional sensor data collection systems [13,28,26] store users' data in a centralized server. Although the centralized approach is simple and straightforward, it has several disadvantages in terms of privacy. First, data is stored in the server, which users may not trust. Second, when the centralized server is compromised, every user's data on the server is breached at the same time. Moreover, as mentioned in Section 1, the IRB regulation requires each institution stores its own data so we need multiple institutional servers. By having remote data stores, SensorSafe supports multiple institutional servers as well as personal data storage that users can trust. The storage can be a personal computer in a user's house, or the institution that collects data can provide a virtual machine pool of individual data stores and make each virtual machine accessible by its owner only. This approach allows users to store data on their own server and reduces the risk of server compromise. The advantages of virtual individual servers are also discussed in [11,34].

Context-Aware Fine-Grained Access Control. Each remote data store provides an access control mechanism. Continuously collected sensor data contain considerable amount of privacy-sensitive information about the data contributors themselves. Location and timestamp information can reveal presence in sensitive places or even life patterns. Sensors such as accelerometers, ECGs, and respiration sensors can tell a lot about contributors' activities and physiological status. Especially when sophisticated inferences are performed, more sensitive information can be revealed such as stress, smoking, conversation [31], or transportation modes [33]. Although data contributors voluntarily participate to share their data, sharing too much information increases their privacy concerns. This concern further increases when contributors understand what kinds of inferences can be drawn from their sensor data [32]. Therefore, we need a mechanism which enables contributors to share only what they want.

In order to support a variety of privacy preferences, our access control mechanism provides various conditions such as data consumer, location, time, sensor, and context. Based on the conditions, contributors can specify actions such as allow, deny, or modify the level of data abstraction. The context condition allows contributors to define privacy rules such as "don't share any data while I am driving." or "don't share data while I am in conversation." Using the conditions and actions, data contributors define a set of rules which express their privacy preferences and the remote data store enforces the rules. Table 1 summarizes the conditions and actions of privacy rules.

Table 1. Various Options for Privacy Rules

(a) Conditions and Actions

Options		Attributes
Conditions	Data Consumer	User Name, Group Name, Study Name
	Location	Pre-defined Label, Region Coordinates
	Time	Time Range, Repeated Time
	Sensor	Sensor Channel Name (e.g., Accelerometer, ECG)
	Context	Available context from sensors (e.g., Moving, Not Moving, Still, Walk, Run, Bike, Drive, Stress, Conversation, Smoke)
Actions		Allow, Deny, Abstraction

(b) Example Abstraction Options

Context	Options
Location	Coordinates, Street Address, Zipcode, City, State, Country, Not Share
Time	Milliseconds, Hour, Day, Month, Year, Not Share
Activity	Accelerometer Data, Still/Walk/Run/Bike/Drive, Move/Not Move, Not Share
Stress	ECG/Respiration Data, Stressed/Not Stressed, Not Share
Smoking	Respiration Data, Smoking/Not Smoking, Not Share
Conversation	Microphone/Respiration Data, Conversation/Not Conversation, Not Share

Basic conditions: Using the data consumer condition, contributors can specify who will be affected by this privacy rule. It can be a unique user name of a data consumer, or a group or study name which includes a set of data consumers. Data contributors specify locations by defining a region on a map user interface. Time condition is defined as a continuous time range (e.g., from Feb. 2011 to Mar. 2011) or repeated time (e.g., 3-6pm on every Wednesday). Using the sensor condition, contributors can select specific sensor channels in their privacy rules.

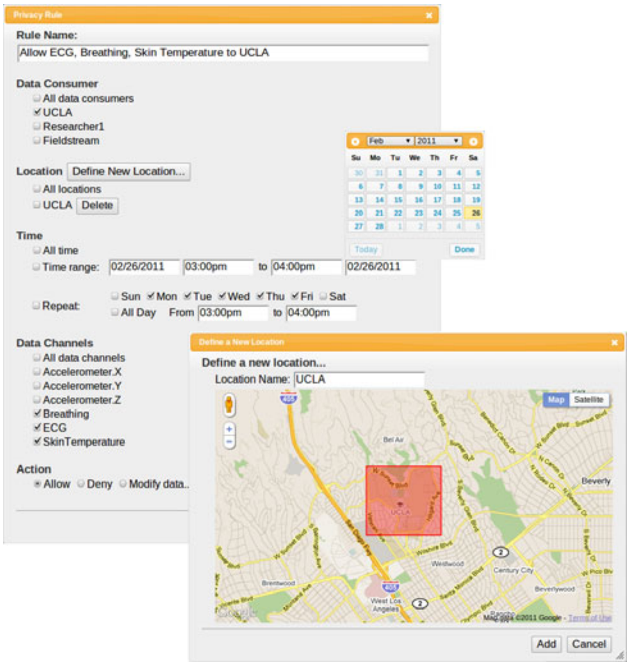
Context condition: Data contributors can also define their privacy rules using context information drawn from sensor data. For example, microphones and respiration sensors can be used to infer whether a data contributor is in conversation or not. An accelerometer with GPS can provide transportation information such as walking, running, biking, and driving. A data contributor might not feel comfortable sharing sensor data while in conversation with someone or while driving. In these cases, context conditions can be used to describe such privacy rules.

Actions: Data contributors can either allow or deny access to sensor data which satisfy the basic and context conditions. When allowed, raw sensor data are shared with corresponding data consumers. In addition, contributors can share more abstracted information instead of sharing raw sensor data. For example, instead of sharing latitude and longitude coordinates, contributors can abstract this information as zip code, city, or state names. With accelerometers, contributors can choose to share only transportation modes (e.g., still, walk, run, bike, drive) or just whether moving or not moving. Note that a sensor can be used to infer multiple context information (e.g., a respiration sensor is used for stress, conversation, and smoking). Therefore, if a contributor chooses not to share such a sensor or a related context, the raw sensor data will not be shared even though other relevant contexts are chosen to be shared in raw data form. For example, if the smoking context is not shared, respiration sensor data will not be shared even though stress and conversation are shared in raw data form. This is because once respiration data are provided by stress or conversation, smoking can be also inferred from the data. The privacy rule processing module contains this sensor/context dependency information and performs access control accordingly.

Privacy rules are created and edited using a web user interface shown in Figure 3. The user interface consists of Google Maps, calendars, dialog boxes, and common HTML UI components such as text boxes, check boxes and radio buttons. Although a usability study of our user interface remains as future work, we believe users will find it easy to create privacy rules because most Internet users are familiar with our web user interface components. Once data contributors define their privacy rules using the web UI, they are stored as JSON objects [4] on the remote data stores. Figure 4 shows an example privacy rule with its corresponding JSON representation.

Data Storage. *Wave Segments:* Another important aspect of a remote data store is that it needs to handle large volumes of data generated by continuous sensing. Storing the time series of sensor data as individual tuples is inefficient both in terms of storage size and querying time. Therefore, in order to increase scalability and computational efficiency, it is essential for a remote data store to have a compact representation of data.

In a remote data store, a continuous stream of sensor data is divided into many segments, called *wave segments*, an extension of an abstract data type proposed in [18]. A wave segment is the smallest unit of data representation, and each wave segment typically contains hundreds or thousands of data samples. We further observe that sensors are typically sampled in uniform intervals, so a wave segment stores timestamp information as a start time and a sampling interval. A wave segment can also have an individual timestamp for each data sample, which is necessary to represent sampling schemes such as adaptive [21], compressive [12], and episodic.



Rule Name: Allow ECG, Breathing, Skin Temperature to UCLA

Data Consumer
- All data consumers
- ✓ UCLA
- Researcher1
- Fieldstream

Location: Define New Location...
- All locations
- UCLA Delete

Time
- All time
- Time range: 02/26/2011 03:00pm to 04:00pm 02/26/2011
- Repeat: Sun ✓ Mon ✓ Tue ✓ Wed ✓ Thu ✓ Fri Sat; All Day From 03:00pm to 04:00pm

Su	Mo	Tu	We	Th	Fr	Sa
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	1	2	3	4	5

Feb 2011

Data Channels
- All data channels
- Accelerometer.X
- Accelerometer.Y
- Accelerometer.Z
- ✓ Breathing
- ✓ ECG
- ✓ SkinTemperature

Action: [✓ Allow] Deny Modify data..

Define a new location...
Location Name: UCLA

Fig. 3. Web User Interface for Privacy Rules

Sequences of data samples from multiple sensor channels are typically stored as Binary Large Objects (blob) in database systems. A wave segment also consists of a sensor value blob and additional metadata describing the value blob. The value blob is an array of tuples each containing values from multiple sensor channels. The metadata includes a start time, a sampling interval, a location, and a format of tuples in the value blob. For non-periodic sampling and mobile sensors, time and location stamps are stored in the value blob as additional sensor channels. Figure 5 shows an example of a wave segment represented in Java Script Object Notation (JSON) [4].

Wave Segment Optimization: The number of wave segments directly affects query performance because it is the number of records stored in a database. Therefore, it is important for each wave segment to include a large enough number of samples. Because memory space is constrained at the sensors, a single data packet from the sensors typically contains dozens or hundreds of samples. For example, the Zephyr chest-band transmits 64 ECG samples in a single packet [7]. If this packet is directly converted to a wave segment, there will be too many wave segments in total decreasing the query performance. Therefore, remote data stores perform a wave segment optimization by merging them as much as possible. If timestamps of two wave segments are consecutive, they can be merged as long as they have the same location coordinates and data channels.

```
[{  'Consumer': [ 'Bob' ],
    'LocationLabel': [ 'UCLA' ],
    'Action': 'Allow'
},
{   'Consumer': [ 'Bob' ],
    'LocationLabel': [ 'UCLA' ],
    'RepeatTime': { 'Day': [ 'Mon', 'Tue', 'Wed', 'Thu', 'Fri' ],
                    'HourMin': [ '9:00am', '6:00pm' ] },
    'Context': [ 'Conversation' ]
    'Action': { 'Abstraction': { 'Stress': 'NotShared' } }
}]
```

Fig. 4. Example of JSON Privacy Rule: "Share all data collected at UCLA with Bob but do not share stress information while I am in conversation at UCLA on Weekdays from 9am to 6pm"

```
{ 'StartTime': null,
  'SamplingInterval': null,
  'StaticLocation': null,
  'ValueBlobFormat': [ 'Time', 'X', 'Y', 'Z',
                       'AccX', 'AccY', 'AccZ' ],
  'ValueBlob': [
     [ 1267662001.752, -118.44304025173187,
       34.069381356239319, null, 1899, 1993, 1614 ],
     [ 1267662001.754, -118.44304025173187,
       34.069381356239319, null, 1900, 1985, 1617 ],
     [ 1267662001.756, -118.44304025173187,
       34.069381356239319, null, 1898, 1990, 1621 ],
     ...
  ]
}
```

Fig. 5. Example of a Wave Segment in Java Script Object Notation

5.2 Broker

In scientific behavioral studies, data consumers (study coordinators such as researchers or doctors) typically recruit many data contributors (study participants) [1,31]. In participatory sensing [10], a data collection campaign also involves many data contributors. In centralized systems, it is trivial to manage those multiple data contributors because every data contributor is registered on a single server. However, if storage for the data contributors are distributed and there is no dedicated server to maintain the list of the data contributors, it is not trivial to manage all the individual data stores.

Data Contributor Management. Therefore, SensorSafe has a broker that manages all the remote data stores so the data consumers can easily access them. The broker stores every data contributor's identity and the IP address of the associated remote data store. Using the web user interface on the broker, the data consumers can create a list of data contributors or search for suitable data contributors. Data contributor searching is further discussed in the following section. The broker also provides a convenient web user interface for accessing contributors' data. The web interface provides query options such as location, time, and data channels so the data consumers can retrieve data in which they are interested. Data consumer applications also can obtain a list of data contributors and their IP addresses by using an API provided by the broker.

Data Contributor Searching. From the data consumer's point of view, a data contributor's privacy rules directly affect the utility of the sensor data. Depending on the privacy rules, a contributor's data could be partially useful for data consumers or not useful at all. For example, suppose a data consumer is interested in studying stress events and related physiological signals in work environments. However, a data contributor participating in the study defines a privacy rule saying he/she does not want to share stress-related data at work place. In this case, these data are not useful to the data consumer. The data consumer needs to find other contributors who share enough data for the data consumer's study.

In SensorSafe, the broker provides a web user interface for searching for data contributors with suitable privacy rules so that data consumers can find contributors who share enough data for their study. The broker locally stores all privacy rules of every user on remote data stores to search through them. Whenever data contributors change their privacy rules, remote data stores automatically communicate with the broker to synchronize the privacy rules. Data consumers can search for all conditions and actions of privacy rules such as location, time, sensor, context, and abstraction. For example, finding data contributors who share ECG and respiration sensor data at the location labeled "work" from 9am to 6pm on weekdays can be performed. After searching suitable data contributors, data consumers can store the list of contributors to access their data.

5.3 Privacy Rule-Aware Data Collection

If a privacy rule says not to share data at a certain location, time, or context, it is better not to collect such data in the first place because the data will not be shared anyway. In order to enable this, smartphones carried by data contributors download the owner's privacy rules from the remote data stores and determine whether to collect data based on the privacy rules. The decision can be made on three conditions such as current location, time, and context. When there are no data to be shared at the current location and time, sensors will be disabled. In case of a context condition, sensor data are first temporarily collected on a smartphone to infer current context. If there are no data to be shared in the current context, the data will be discarded.

Although privacy rule-aware data collection provides a more secure way to collect data, but one should not overlook cases in which a data contributor wants to change privacy rules after collecting data. If a contributor wants to share data that have not been collected at all, there is no way to recover them. Therefore, we provide privacy rule-aware data collection as optional functionality, and data contributors need to carefully decide whether or not to use this option.

5.4 Authentication

When data consumers and contributors access the broker and remote data stores through APIs, they are authenticated by their unique API keys. An API key is a random string generated by the SHA algorithm. Each user obtains a unique API key when he/she is first registered to the servers. Users must keep their API keys private because it acts as a username and a password. In order to secure the API key during communications, it is included in the body of a HTTPS POST request and the communication is secured with HTTPS. When a data consumer first accesses a certain data contributor's remote data store, he/she needs to register to the remote data store and obtain an API key. Therefore, a data consumer might have many API keys for multiple remote data stores. However, the registration process is automatically handled by the broker and the list of API keys are stored on the broker. Data consumer applications use the HTTP API on the broker to obtain a list of data contributors with corresponding remote data stores and API keys. Accesses to web user interfaces are authenticated by a login system using a username and a password.

6 Application Examples

In this section, we show how SensorSafe can support sharing sensor data in privacy-preserving way. Our example scenarios include two applications: a medical behavioral study and a health-care application. In the behavioral study, researchers want to analyze effects of various environmental factors on an individual's stress level [31]. For this study, data contributors wear a chest band equipped with an ECG and a respiration sensor. They also carry smartphones which record acceleration, time, location, and voice on the microphone. They live their normal life as the sensor data are collected automatically. On the smartphone, various contextual information such as stress, smoking, conversation, and transportation modes are inferred using the sensors on the phone and the chest band. The sensor data are annotated with the context information and uploaded to remote data stores. In our health-care application scenario, data contributors want to share their daily activities with personal coaches to get advice on exercise and health habit [6].

A data contributor, say Alice, first decides to share all data with the researchers. After logging into her remote data store, she defines a privacy rule that allows the researchers to access all the data. She also thinks her health coach only needs activity data so she defines a privacy rule that allows the health coach to access accelerometer data only. After collecting data for one day, Alice reviews her data using the web user interface on her data store. Alice finds out that she is frequently stressed while driving. She feels uncomfortable with sharing this information so she adds a privacy rule that denies access to stress data while driving. She also feels uncomfortable sharing activity data while she is home so she adds a privacy rule which denies accelerometer data collected at her home location. She is certain that she is not going to change her mind about sharing her data so she turns on privacy rule-aware data collection on her smartphone. Whenever the smartphone detects she is driving, it stops collecting ECG and respiration data which are related to stress inference. Whenever the smartphone detects that current location is her home, it also stops collecting accelerometer data.

A data consumer, say Bob, wants to obtain some data for his study. He has recruited 20 data contributors including Alice. He first logs into the broker server and adds the data contributors to his account. When he adds his data contributors, the broker automatically registers Bob to the remote data stores to obtain an API key. Bob is especially interested in people's stress behavior while they are driving. Because Bob knows that not every data contributor will share their stress information while driving, he uses a data contributor searching function on the broker. After searching for suitable data contributors, he obtains a list of data contributors without Alice and saves the list in his account. Bob reviews the contributors' data using the web user interface provided on the broker. Bob also uses his own data analyzing software. The software first obtains the list of data contributors with access information for their remote data stores from the broker. Then, the software downloads the contributors' data using the query API provided by each remote data store.

7 Conclusion and Future Work

This paper presents SensorSafe, a framework that enables data contributors to share their personal sensory information with data consumers in privacy-preserving way. Using its context-aware fine-grained access control mechanism, data contributors can define sharing rules with various conditions including current context or behavior. To improve the usability of SensorSafe, we implemented web-based user interfaces for defining privacy rules and reviewing a user's data. In addition, data contributors retain the ownership of their data by using remote data stores. SensorSafe also supports applications involving data collection from multiple contributors and institutions by having a separate broker server. We provide application examples to show how SensorSafe can support user privacy and management of sensory information. In the future, user studies will be conducted to evaluate and improve the usability of our system and also provide understanding of how people share personal data with others. Moreover, in order to improve security of the SensorSafe architecture, we will analyze our system for various attack scenarios and implement appropriate security mechanisms.

References

1. FieldStream: network data services for exposure biology studies in natural environments, http://www.fieldstream.org/
2. Google health, http://www.google.com/intl/en/health/about/
3. Institutional Review Board - Protect Research Data, http://irb.ufl.edu/irb01/data.html
4. JavaScript object notation, http://www.json.org/
5. Microsoft HealthVault, http://www.healthvault.com

6. Philips DirectLife: fitness, health and successful weight management, http://www.directlife.philips.com/
7. Zephyr technology corporation, BioHarness BT, http://www.zephyr-technology.com/bioharness-bt
8. Ahmadi, H., Pham, N., Ganti, R., Abdelzaher, T., Nath, S., Han, J.: Privacy-aware regression modeling of participatory sensing data. In: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, pp. 99–112 (2010)
9. Baden, R., Bender, A., Spring, N., Bhattacharjee, B., Starin, D.: Persona: an online social network with user-defined privacy. ACM SIGCOMM Computer Communication Review 39(4), 135–146 (2009)
10. Burke, J., Estrin, D., Hansen, M., Parker, A., Ramanathan, N., Reddy, S., Srivastava, M.: Participatory sensing. In: World Sensor Web Workshop, pp. 1–5 (2006)
11. Cáceres, R., Cox, L., Lim, H., Shakimov, A., Varshavsky, A.: Virtual individual servers as privacy-preserving proxies for mobile devices. In: Proc. of the 1st ACM Workshop on Networking, Systems, and Applications for Mobile Handhelds, pp. 37–42 (2009)
12. Candes, E., Romberg, J., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Communications on Pure and Applied Mathematics 59(8), 1207–1223 (2006)
13. Chang, K., Yau, N., Hansen, M., Estrin, D.: Sensorbase.org: a centralized repository to slog sensor network data. In: Proc. of the International Conf. on Distributed Networks (DCOSS)/EAWMS (2006)
14. Cornelius, C., Kapadia, A., Kotz, D., Peebles, D., Shin, M., Triandopoulos, N.: AnonySense: Privacy-aware people-centric sensing. In: Proc. of the 6th International Conference on Mobile Systems, Applications, and Services, pp. 211–224 (2008)
15. Cornwell, J., Fette, I., Hsieh, G., Prabaker, M., Rao, J., Tang, K., Vaniea, K., Bauer, L., Cranor, L., Hong, J., et al.: User-controllable security and privacy for pervasive computing. In: Proc. of the 8th IEEE Workshop on Mobile Computing Systems and Applications, pp. 14–19 (2007)
16. Fung, B.C., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey on recent developments. ACM Computing Surveys (2010)
17. Ganti, R., Pham, N., Tsai, Y., Abdelzaher, T.: PoolView: stream privacy for grassroots participatory sensing. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems, pp. 281–294 (2008)
18. Girod, L., Mei, Y., Newton, R., Rost, S., Thiagarajan, A., Balakrishnan, H., Madden, S.: XStream: a Signal-Oriented Data Stream Management System. In: Proc. of IEEE 24th International Conference on Data Engineering, pp. 1180–1189 (2008)
19. Hansell, S.: AOL removes search data on vast group of web users. New York Times (August 8, 2006)
20. Hoh, B., Gruteser, M., Herring, R., Ban, J., Work, D., Herrera, J., Bayen, A., Annavaram, M., Jacobson, Q.: Virtual trip lines for distributed privacy-preserving traffic monitoring. In: Proc. of the 6th International Conference on Mobile Systems, Applications, and Services, pp. 17–20 (2008)
21. Jain, A., Chang, E.: Adaptive sampling for sensor networks. In: Proc. of the 1st International Workshop on Data Management for Sensor Networks: in Conjunction with VLDB 2004, pp. 10–16 (2004)
22. Kotz, D., Avancha, S., Baxi, A.: A privacy framework for mobile health and home-care systems. In: Proc. of the first ACM Workshop on Security and Privacy in Medical and Home-care Systems, pp. 1–12 (2009)

23. Krumm, J.: Inference attacks on location tracks. In: LaMarca, A., Langheinrich, M., Truong, K.N. (eds.) Pervasive 2007. LNCS, vol. 4480, pp. 127–143. Springer, Heidelberg (2007)
24. Li, N., Li, T., Venkatasubramanian, S.: t-Closeness: privacy beyond k-Anonymity and l-Diversity. In: Proc. of IEEE 23rd International Conference on Data Engineering, pp. 106–115 (2007)
25. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (March 2007)
26. Miluzzo, E., Lane, N., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S., Zheng, X., Campbell, A.: Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In: Proc. of the 6th ACM Conference on Embedded Network Sensor Systems, pp. 337–350 (2008)
27. Mun, M., Hao, S., Mishra, N., Shilton, K., Burke, J., Estrin, D., Hansen, M., Govindan, R.: Personal data vaults: a locus of control for personal data streams. In: Proc. of the 6th International Conference, p. 17 (2010)
28. Mun, M., Reddy, S., Shilton, K., Yau, N., Burke, J., Estrin, D., Hansen, M., Howard, E., West, R., Boda, P.: PEIR, the personal environmental impact report, as a platform for participatory sensing systems research. In: Proc. of 7th International Conference on Mobile Systems, Applications, and Services, pp. 55–68 (2009)
29. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: Proc. of IEEE Symposium on Security and Privacy, pp. 111–125 (2008)
30. Narayanan, A., Shmatikov, V.: De-anonymizing social networks. In: Proc. of IEEE Symposium on Security and Privacy, pp. 173–187 (2009)
31. Plarre, K., Raij, A., Hossain, S.M., Ali, A.A., Nakajima, M., al'Absi, M., Ertin, E., Kamarck, T., Kumar, S., Scott, M., Siewiorek, D., Smailagic, A., Wittmers Jr., L.E.: Continuous inference of psychological stress from sensory measurements collected in the natural environment. In: Proc. of 10th International Conference on Information Processing in Sensor Networks (2011)
32. Raij, A., Ghosh, A., Kumar, S., Srivastava, M.: Privacy risks emerging from the adoption of innocuouswearable sensors in the mobile environment. In: Proc. of ACM CHI Conference on Human Factors in Computing Systems (2011)
33. Reddy, S., Burke, J., Estrin, D., Hansen, M., Srivastava, M.: Determining transportation mode on mobile phones. In: Proc. of 12th IEEE International Symposium on Wearable Computers, pp. 25–28 (2008)
34. Seong, S., Seo, J., Nasielski, M., Sengupta, D., Hangal, S., Teh, S., Chu, R., Dodson, B., Lam, M.: PrPl: a decentralized social networking infrastructure. In: Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, pp. 1–8 (2010)
35. Sweeney, L.: k-anonymity: a model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10, 557–570 (2002)
36. Toch, E., Cranshaw, J., Hankes-Drielsma, P., Springfield, J., Kelley, P.G., Cranor, L., Hong, J., Sadeh, N.: Locaccino: a privacy-centric location sharing application. In: Proc. of 12th ACM International Conference on Ubiquitous Computing, pp. 381–382 (2010)
37. Tootoonchian, A., Saroiu, S., Ganjali, Y., Wolman, A.: Lockr: better privacy for social networks. In: Proc. of the 5th International Conference on Emerging Networking Experiments and Technologies, pp. 169–180 (2009)
38. Wikipedia. Personal health record, http://en.wikipedia.org/wiki/Personal_health_record

Watermarking for Adaptive Streaming Protocols

Dmitri Jarnikov[1,2] and Jeroen M. Doumen[1]

[1] Irdeto B.V., The Netherlands
{djarnikov,jdoumen}@irdeto.com
[2] Eindhoven University of Technology, The Netherlands
d.s.jarnikov@tue.nl

Abstract. Online multimedia distribution is often done by means of adaptive streaming protocols. To protect this content, its owners apply a unique watermark to each copy. However, sending a unique copy to each client incurs a prohibitive cost, especially in terms of bandwidth and server load, and embedding a watermark on a client device not only compromises the system security, but also is not feasible in the uncontrolled environment such as the Internet. In this paper, we propose to solve this problem by creating a few streams, each with different but constant watermarks, and force the client to switch between the streams. This will result in a uniquely watermarked stream for the each client. We illustrate our solution on the example of the currently deployed adaptive streaming protocols.

Keywords: watermarking framework, adaptive streaming, anti-piracy.

1 Introduction

Today many people explore the possibilities for consuming media over the Internet. The majority of solutions for multimedia delivery that are on the market today use real-time streaming. The advantages of the real-time streaming – real-time content delivery (live events can be viewed as they happen) and the ability to play content as soon as transmission is started barely balance the fact that it puts a big burden on content providers due to the need for a streaming server, sophisticated techniques to deal with users' bandwidth constraints and special ways of traversing firewalls [14]. One of the biggest disadvantages of traditional streaming is that it does not scale, since, in practice, it is based on a unicast protocol (RTP/UDP [19] or proprietary protocols).

Recently, a new generation of technologies that employ progressive download has emerged with the aim to overcome this weakness of streaming while providing a similar quality of experience to end users. To access content via progressive download a user-device uses HTTP/TCP [8] to request (typically consecutive) parts of content from the server. The new technology enables solutions that require only a standard web server, provide a guaranteed high-quality delivery with no lost or discarded data and easily cross most firewalls. Moreover, use of HTTP allows the solutions to benefit from proxy caching that reduces transmission latency and decreases server/network loads. The main disadvantages of

© Springer-Verlag Berlin Heidelberg 2011

progressive download – potentially slower delivery time due to packet retransmission and paying no regards to the user's bandwidth constraints are being addressed by adaptive streaming protocols [18,17].

The latest developments of progressive download for delivering multimedia content spawned protocols that allow transmitting content as a set of time-bounded segments called chunks. The protocols starts by having a client download a description file (also known as "manifest" or "playlist") that lists the location(s) of content chunks and describes rules for forming a request to access a chunk. To consume content, the client first obtains the description file and then requests, gets and consumes content chunks. Since each chunk has a unique address, the protocols support web caching. So, if multiple clients request the same chunk from the adaptive streaming server, there is a high probability that some of them will receive the chunk from the cache of an intermediate web node, instead of from the server itself. Using web caches enables building systems that can easily scale from hundreds of users to millions.

Chunks can be separated physically (a separate file is created for each chunk of content) or logically (all chunks of content are stored in a single file with an addressing structure that allows accessing any chunk individually). Protocols that are based on physically separated chunks break the content into a sequence of short chunks of equal duration. At the start of the streaming session, the client downloads a description file that contains an ordered list of URIs referring to media files (each media file contains a single chunk of content) that the client may consume. Protocols that are based on logically separated chunks relay on a file format that allows addressing parts of the content at a number of predefined access points (time offsets). At the start of the streaming session, the client downloads a description file that contains the name of the media file and rules to create a URI that points to the files with a given offset. Physical separation is used by HTTP Live Streaming [18] and 3GPP adaptive HTTP Streaming [6], whereas logical separation is employed by Microsoft SmoothStreaming [17].

The basic progressive download can be extended with support for the delivery of content over a network infrastructure that has no quality-of-service guarantees. This is done by providing multiple copies of the same content with different bitrates, spatial resolution and/or other encoding characteristics (these copies are also called quality levels). Since each quality level (copy) consists of a sequence of time-aligned chunks, a client can switch between several quality levels at run-time on a chunk-by-chunk basis to react to varying transmission and/or processing conditions. Due to its adaptive nature, this new class of protocols is called "adaptive streaming protocols".

A number of challenges exist, however, when using adaptive streaming protocols for delivering premium content over the Internet. One challenge is to prevent illegal copying and distribution of content. In addition to the attacks that are used against traditional streaming (capturing still frames from monitors, recording video data traffic on the network interface, replaying connection requests, data transfer capturing, etc.), protocols based on progressive download may allow retrieving video parts from the browser cache directly and, after merging the parts together, redistributing content. A commonly used counter mechanism to such attacks (except for screen capturing) is the protection of the content by a DRM/CA system. From the consumer's perspective, however, a content protection system that does not block all actions that may, potentially, lead to an infringement, but that offers possibilities to trace the pirate once an infringement occurred is preferable. Such protection systems can be implemented using watermarking technology. Watermarking is a process of including a hidden signal (watermark) in the content in a non-removable manner.

Typically, to allow unambiguous identification of every user of a system, the server must uniquely watermark each item of content for each user or entity to which the content is to be distributed. If the number of users that consume content is large (which is a typical case for Internet content delivery solutions that are the main adopters of the progressive-download protocols), watermarking can be computationally intensive for the server. Moreover, since protocols as described above rely on web caches to achieve scalability, reduce latency and to minimize server load, it is not possible to maintain the performance of the web caching if the content chunk changes are unique for every client device. As adaptive streaming protocols depend on web cache for efficiency benefits, unique content chunks are a disadvantage. Finally, any watermarking solution should be directly applicable to existing adaptive streaming protocols without any changes to the protocols and, if possible, to existing client devices.

This paper describes the way to uniquely watermark content distributed by adaptive streaming protocols. The approach is to convert each chunk from the original content into two (or more) chunk instances, each with a different content modification (i.e. bearing a different watermark). Then, for protocols that rely on physical separation of chunks, the server generates a personalized playlist consisting of a unique sequence of modified chunks, so that the resulting content is uniquely watermarked. For protocols that use logical chunks, additional client-based software is proposed that enables watermarking by manipulating the HTTP requests from the client device to the web server.

2 Solution Description

The proposed solution creates a limited number of watermarked copies of content on the server and assembles a watermark that is unique for each user by reusing mechanisms of chunk-based transmission technologies. Generally speaking, any adaptive streaming solution consists of two parts: a server and a client. The server is responsible for encoding, encapsulating and preparing content for distribution as well as for supporting a standard web server functionality. The client is responsible for fetching description data from the server, parsing the description to make a list of content chunks to be downloaded, and acquiring and presenting the chunks in a continuous way to the user. In the proposed solution, the server undergoes the most modifications, whereas the client is not modified at all (for physically separated chunks) or modified slightly (for logically separated chunks).

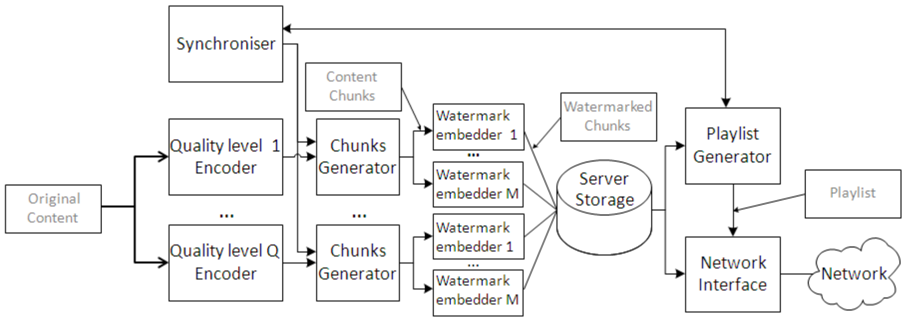


Fig. 1. Server architecture diagram

The server architecture diagram (Figure 1) shows Q quality level encoders to transcode/encode the original content into a set of derived content versions at different quality levels $\{q_1, \ldots, q_Q\}$. Each derived content version is then split by the chunk generators into N time-aligned chunks $\{c_1, \ldots, c_N\}$. The duration of the chunks is shared between the synchroniser and the playlist generator that creates the playlist on the basis of information in the chunk store[1]. The watermark embedder creates M copies of each chunk. We define the watermark length l such that we have enough symbols (e.g. bits) to identify each client uniquely (plus, optionally carry some service information). The watermark symbol length is chosen equal to the length of a chunk. Each copy of chunk c_k (of quality level q_j), $m_i.q_j.c_k$, is embedded with its own watermark symbol w_i. So, the modified chunks $m_i.q_j.c_k$ (for $i = 1 \ldots M$) contain M different watermark symbols, even though they contain the same piece of content. Generally, there will be more chunks then the watermark length, so we embed the watermark repeatedly.

The most important part of the solution is to make the client request chunks based on its identity in such a way that after transmission of l chunks each client has a unique combination of modifications in its received set $m_i.q_j.c_k$. In this fashion, the content that is transmitted to each client is uniquely watermarked. For example, in a system with two quality levels and two different watermarks, one can divide the chunks into subsets by the different modifications: the set S_1 of all copies watermarked with w_1 consists of $S_1 = \{m_1.q_1.c_k\} \cup \{m_1.q_2.c_k\}$. Similarly, we have $S_2 = \{m_2.q_1.c_k\} \cup \{m_2.q_2.c_k\}$ as the set of chunks watermarked with w_2. After transmission of l chunks each client has a unique combination of chunks from subsets S_1 and S_2. Thus, the content that is transmitted to the client is uniquely watermarked.

In the case of the physically separated chunks that are prepared as described above, the server creates a unique description file based on the client's identity and transmits it to the said client. Figure 2 shows an example of playlist generated for a user with ID 0100 in a system with two quality levels (q_1 and q_2), four chunks ($c_1, \ldots, c_4$) and two watermarks (w_1 and w_2) that have at least four

[1] Note that chunk creation may occur before quality level encoding.

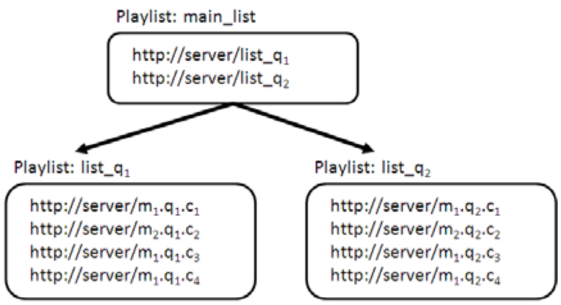


Fig. 2. Example playlists for physically separated chunks watermarked with symbols w_1, w_2, w_1, and w_1 respectively

symbols. The main playlist tells the client that two quality levels are available and the playlists for these levels are $list\_q_1$ and $list\_q_2$. The latter playlists provide URIs for every chunk taking chunk modifications that correspond to the user ID (for 0100 these are $m_1 m_2 m_1 m_1$). Even if the client randomly switches between $list\_q_1$ and $list\_q_2$ but keeps the order of chunks intact (i.e. chunk order c_1, c_2, c_3, c_4), its content will be correctly watermarked. Thus, by making a unique sequence of modified content chunks, the adaptive streaming system constructs a unique content file for each particular streaming content receiver.

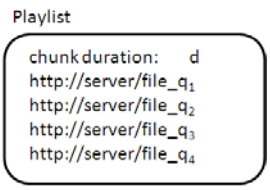


Fig. 3. An example playlists for the logically separated chunks

The described approach is not possible for solutions based on logically separated chunks, since the description file that the server deliver to clients contains the address of a single file per quality level (e.g. $file\_q_1$, $file\_q_2$,..., $file\_q_n$). Figure 3 shows an example of a playlist generated in a system with four quality levels (q_1 through q_4) and with chunks of duration d. The client request content from the server by forming URI requests of the form $http://server/file\_q_j@t_k$, where q_j is the chosen quality level and t_k is the time offset at step k.

Even though the modifications of the content are still performed in accordance to the diagram shown in Figure 1 and the server storage contains modifications $m_1.file\_q_j$ and $m_2.file\_q_j$ of the derived content version $file\_q_j$, it is neither possible nor desirable (from a security perspective) to refer to both modifications in the same playlist.

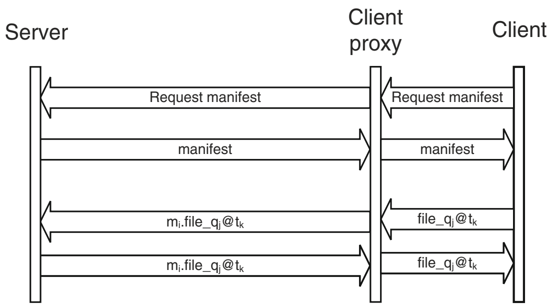


Fig. 4. Illustration of chunk requests modification for the logically separated chunks. The user-specific information is the modification version m_i (m_i is modification version at step i) that is inserted before the file name.

The only feasible solution found is to either modify the client to include user-specific information into the URIs that would allow the server to redirect the requests to the particular modification of the file, or to use an additional 'proxy' component on the user device that intercepts HTTP requests from the adaptive streaming client and alters them with user-specific information (Figure 4). In both cases the client side will be responsible for initiating the customization of the watermark even though the customization itself will still be done on the server.

The playlists (Figure 2) and user-specific information (Figure 4) in the examples above use a structured naming sequence for modified content chunks. This would allow an attacker to make educated guesses for content chunks that are modified differently and create content in a way that bypasses the tracking mechanism. Hence, an attacker should be unable to guess which modified chunks contain the same content. This can be achieved by applying a pseudo-random permutation, as discussed in the section below.

3 References Randomization

In the first step content preparation, chunk creation and chunk modification are performed as described earlier. Figure 5 depicts this for two different quality settings (q_1 and q_2) and two different modifications (m_1 and m_2). The second step assigns a unique number to each chunk. For instance, a chunk with label $m_i.q_j.c_k$ can be assigned number $n(i, j, k) = i + M * j + M * Q * k$. Note that at this stage, it is still easy for an attacker to guess which chunk numbers contain the same content. For example, in Figure 5, chunks 0 through 3 contain the same content in different quality levels and with different modifications. In the third step, each chunk is assigned a (pseudo)random number. Renumbering is done by selecting a pseudorandom permutation σ on the range $[1, \ldots, M * Q * N]$, and applying this to the unique number of each chunk. For example, in Figure 5 we have that $\sigma(0) = 55$ and $\sigma(1) = 12$. In this way, an attacker cannot predict

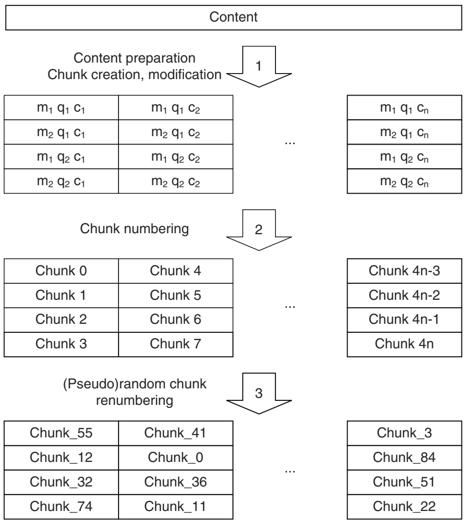


Fig. 5. Example permutation

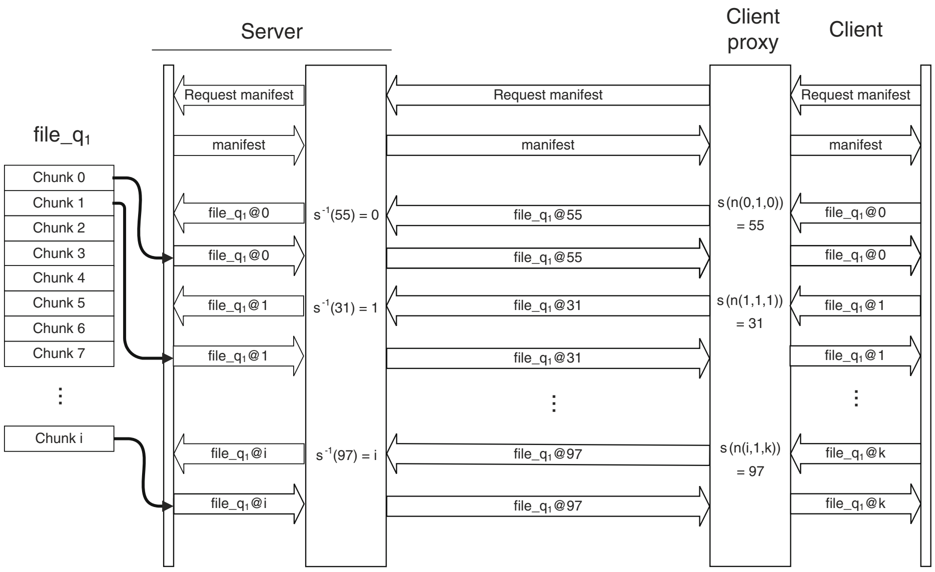


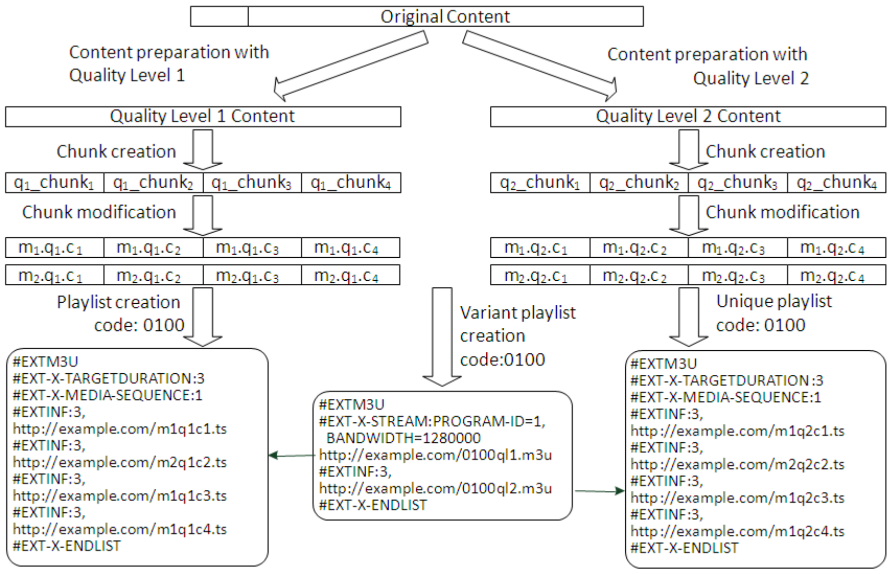
Fig. 6. Streaming from the server to the client with ID ’0100’

which chunks contain the same piece of content. One can make the permutation σ unique for each piece of (original) content by seeding a pseudorandom permutation generator with e.g. the original filename.

The algorithm discussed above can be directly applied to solutions based on physically separated chunks, whereas other solutions require some additional implementation. For solutions based on logically separated chunks, the client will be provisioned with a tamper-resistant piece of software (called client 'proxy' in Figure 4) that is able to calculate the permutation σ. This software will have the client ID embedded in the form of modifications m_i it can make. In the above figure, when the client asks for the first chunk of content (the request "file@0"), the proxy translates this, using the permutation σ, into the actual chunk number on the server: $\sigma(n(0,1,0)) = 55$. In particular, as the client ID is embedded, the client proxy does not need to be able to calculate the permutation σ for all possible inputs, just for those that match its expected modifications. For example, if the embedded modification for a particular client will be 0100, the client need only calculate the values $\sigma(n(0,j,4k+1))$, $\sigma(n(1,j,4k+2))$, $\sigma(n(0,j,4k+3))$ and $\sigma(n(0,j,4k+4))$. There is never any need for this client to for instance evaluate $\sigma(n(1,j,4k+1))$, as this would indicate a modification value 1 in the first position. The process of streaming from the server to the client is depicted in Figure 6.

4 HTTP Live Streaming Example

To illustrate application of the proposed solution to an existing adaptive streaming protocol, we have chosen HTTP Live Streaming from Apple[18]. Although it is targeted at Apple devices (e.g. iPhone, iPod, Macintoshes), it can be seen as a typical example of a protocol that is based on physical separation of chunks. In HTTP Live Streaming a continuous stream of content, multiplexed in an MPEG-2 transport stream [11], is divided into a set of short MPEG-2 TS (*.ts*) files with


Original Content
Content preparation with Quality Level 1
Content preparation with Quality Level 2
Quality Level 1 Content
Quality Level 2 Content
Chunk creation
Chunk creation
q1_chunk1 | q1_chunk2 | q1_chunk3 | q1_chunk4
q2_chunk1 | q2_chunk2 | q2_chunk3 | q2_chunk4
Chunk modification
Chunk modification
m1.q1.c1 | m1.q1.c2 | m1.q1.c3 | m1.q1.c4
m2.q1.c1 | m2.q1.c2 | m2.q1.c3 | m2.q1.c4
m1.q2.c1 | m1.q2.c2 | m1.q2.c3 | m1.q2.c4
m2.q2.c1 | m2.q2.c2 | m2.q2.c3 | m2.q2.c4
Playlist creation code: 0100
Variant playlist creation code:0100
Unique playlist code: 0100

```
#EXTM3U
#EXT-X-TARGETDURATION:3
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:3,
http://example.com/m1q1c1.ts
#EXTINF:3,
http://example.com/m2q1c2.ts
#EXTINF:3,
http://example.com/m1q1c3.ts
#EXTINF:3,
http://example.com/m1q1c4.ts
#EXT-X-ENDLIST
```

```
#EXTM3U
#EXT-X-STREAM:PROGRAM-ID=1,
 BANDWIDTH=1280000
http://example.com/0100ql1.m3u
#EXTINF:3,
http://example.com/0100ql2.m3u
#EXT-X-ENDLIST
```

```
#EXTM3U
#EXT-X-TARGETDURATION:3
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:3,
http://example.com/m1q2c1.ts
#EXTINF:3,
http://example.com/m2q2c2.ts
#EXTINF:3,
http://example.com/m1q2c3.ts
#EXTINF:3,
http://example.com/m1q2c4.ts
#EXT-X-ENDLIST
```

Fig. 7. An example of HTTP Live Streaming playlist

each file's URI being stored in a M3U [10] playlist file. The client fetches the playlist file, reads it, requests the listed *.ts* files in order and displays them to the user in the right order.

Figure 7 show an example of HTTP Live Streaming playlist. In the example, the adaptive streaming server prepares a playlist for two quality levels with a unique sequence of modified chunk references to insert 0100 watermark. In the figure, the sequence is inserted by the modified content chunk sequence ($m_1.q_i.c_1$, $m_2.q_i.c_2$, $m_1.q_i.c_3$, $m_1.q_i.c_4$). For each quality level, the appropriate watermarking symbol will be generated in the output of the receiver.

5 Discussion

5.1 Scalability of the Solution

In this section, we discuss the scalability of our solution. The number of users that can be assigned with a unique watermark depends on the number of chunks that we can embed with a watermark symbol (N), and the number of differently watermarked copies of the content (M). In practice, watermarking systems required to support 64 bits of payload [4], which is more than sufficient to uniquely identify users of any existing system. So, even if only two copies of watermarks (i.e. $M = 2$ and the first copy is watermarked with bits 0 while the second copy is watermarked with bits 1) are created, we need 64 chunks to embed the whole watermark. Assuming that the chunk length is 5 seconds, we need 320 seconds of content to embed a watermark.

A larger value of M means that we need lesser chunks to embed the watermark. Higher value of M, however, requires more copies to be created, which means an increased storage demand. It is likely that content providers would choose having just two copies ($M = 2$), which, in turn, pose a question about effectiveness of using 2 to 5 second chunks for embedding just a single bit. Schemes exist (e.g. [15]) that allow to embed large payloads in the given amount of content. An optimization of the proposed solution could be achieved by combining a user-specific payload (i.e. a bit of the watermark that carries the user's ID) with a payload that is specific to the content or content distributer or with a service payload (e.g. synchronization bits, timing bits). Since a the latter payload is the same for all users, it is transparent to switching between chunks with different user-dependent watermark symbols. Figure 8 demonstrates a three-section concept to mix user- dependent and independent (common) watermarks. The figure features one common section before and one after each watermarking section.

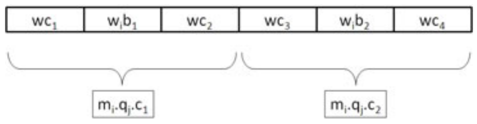


Fig. 8. Example of content watermarked content with w_1 and w_2. w_ib_1 and w_ib_2 are the first and the second bits of watermark w_i. wc_x is x^{th} common section.

The chunk segmentation of the content occurs between two adjustment common sections when the three-section concept is employed.

5.2 Security of the Solution

In this section, we analyze the security weaknesses our scheme might be susceptible to, and suggest defenses against these known attacks. We can divide the possible attacks on our scheme in two broad classes: firstly, an attacker could try to attack the watermarking technology directly. In this case, the attack is performed without regard to our framework's stream switching. We assume that the employed watermarking technology is robust enough to withstand this class of attacks.

Secondly, an attacker can try to mount a collusion attack. In such an attack, he will obtain a number of (uniquely watermarked) copies of the same content, which enables him to compare sections and see the differences. Based on these copies, the attacker will produce a pirated copy that he distributes. Here, the challenge is to still identify (a subset of) the original copies that were used in constructing the pirated copy. This problem is called traitor tracing in the literature, and solved by applying fingerprinting codes.

We will assume the marking assumption applies, as introduced by [3]. This assumption states that if (in a certain chunk) the attacker gets the same watermark symbol in all copies, he is forced to output this same symbol in the pirated copy. As the pirate sees no difference across all his copies, in our scheme this assumption is realistic given the robustness of the underlying watermarking technology.

We choose to use the Tardos fingerprinting code [21] as it has two key benefits: firstly, recent work [20,1,9] has focused on the length of the codewords to be embedded, resulting in constructions with codewords of length $(2 \ln 2) c^2 \ln \epsilon_1^{-1}$ that are designed to resist attackers that obtains up to c different copies, and a probability ϵ_1 of accusing an innocent user. Secondly, the code is frameproof: even if an attacker obtains more copies than the code was designed to resist, the attacker is unable to frame innocent users.

Note that the Tardos fingerprinting code described above is static: each user is assigned a long, fixed (random) string to embed into his copy, and this set of strings is not adapted dynamically after the tracer runs an analysis. One could also employ a dynamic traitor tracing solution [16], which will alleviate many difficulties that Tardos code will introduces. First of all, such a dynamic Tardos code no longer needs to fix the number of colluders beforehand. Secondly, as traitors can be removed immediately after each symbol (at least in the live rebroadcasting scenario) once their behaviour becomes too suspicious, the actual code length needed in practice is smaller than expected. Also, if the available number of different modifications is large enough, Fiat-Tassa type traitor tracing schemes [7] can be employed. We leave the further investigation of dynamic schemes as future work.

In the case the chunks are logically separated, an additional class of attacks exists. As the client proxy has to calculate the pseudorandom permutations by itself, an attacker could try to influence this part of the client proxy software (and for instance try to force it to only select chunks with modifications $m_i = 1$). We can deal with this attack in two ways: as mentioned before, we assume that (this part of) the client proxy is tamper-resistant [12], and that such a change is infeasible. A second defense can be mounted by making sure the proxy software is only capable of generating its own pseudorandom permutations. This is especially delicate when Tardos fingerprinting codes are used, as a priori all modifications are possible, but dependent on the selected bias value. In this case, the client proxy should be provided with an authenticated bias value, which is then used together with its identity (and the chunk index and desired quality level) to generate the correct permutation. For example, one can be achieve this by using whitebox cryptography [5] and hash functions.

To summarize, our watermarking framework is at least as secure as the underlying watermarking technology. We can defend against collusion attacks by employing fingerprinting codes.

6 Conclusions

In conclusion, our solution allows for efficient watermarking of adaptive streaming content, by utilizing the protocols' native property of dividing content into chunks and by using the idea of creating multiple variations of each chunk (at least two alternatives per chunk) that has been described in [23,2,13]. The solution can be applied to the physically and logically separated cases of adaptive streaming.

Having one or more modified chunk instances for each original chunk at a given quality level, increases the total number of chunks and only slightly decreases the effectiveness of the web caching. More precisely, as each modification results in a new, different stream, the cache effectiveness is reduced linearly with the number of modifications.

As for the security of our solution, a user is deterred from distributing either his playlist (in the physically separated case) or his software client (in the logically separated case), as it will be traceable. Solutions based on physically separated chunks are secure from an attacker as the attacker is unable to find other modified versions of the original chunks. In systems with logically separated chunks, a hardened solution that can evaluate the pseudorandom permutation of chunks for the personalised (user-id based) series of modifications (either a separate 'proxy', or integrated into the client) is required.

Attacks that employ the logging and replay of connection requests [22],can also be defeated by making the chunk renumbering permutation change pseudorandomly over time. This might be more efficient than relying on fingerprinting codes to defeat this attack as well. We leave the analysis of employing this additional defense technique as future work.

Employing a dynamic traitor tracing solution will make the tracing more efficient – both dynamic Tardos schemes [16] and Fiat-Tassa type schemes [7] are candidates. As future work, we intend to investigate the efficiency of the tracing procedure.

The proposed solution describes a watermarking application (i.e. when and how content is manipulated to apply a watermarking technology) rather than a watermarking technology (i.e. how data representing the watermark value is hidden in and retrieved from content). As a result, the proposed system is agnostic about the actual watermarking technology used.

References

1. Amiri, E., Tardos, G.: High rate fingerprinting codes and the fingerprinting capacity. In: Mathieu, C. (ed.) SODA, pp. 336–345. SIAM, Philadelphia (2009)
2. Bloom, J.A., Isnardi, M.A., Polyzois, C.A., Hurst Jr., R.N., Tinker, M.: Method and apparatus for creating multiple unique copies of content. U.S. Patent #20050025463 (February 2005).
3. Boneh, D., Shaw, J.: Collusion-secure fingerprinting for digital data. IEEE Transactions on Information Theory 44(5), 1897–1905 (1998)
4. Cheveau, L.: Choosing a watermarking system for digital television - the technology and the compromises. In: IBC 2002 (2002)
5. Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C.: White-box cryptography and an aes implementation. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 250–270. Springer, Heidelberg (2003)
6. Ericsson. 3GPP adaptive HTTP Streaming, AHS (2010)
7. Fiat, A., Tassa, T.: Dynamic traitor tracing. J. Cryptology 14(3), 211–223 (2001)
8. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: RFC 2616, Hypertext Transfer Protocol – HTTP/1.1 (1999)
9. Huang, Y.-W., Moulin, P.: Saddle-point solution of the fingerprinting capacity game under the marking assumption. CoRR, abs/0905.1375 (2009)
10. ID3.org. The ID3 audio file data tagging format (2007)
11. International Organization for Standardization. ISO/IEC 13818-1, Information Technology-Generic Coding of Moving Pictures and Associated Audio, Part 1: Systems. International Organization for Standardization, Geneva, Switzerland (1994)
12. Irdeto Cloakware. Introduction to Application Security
13. Jin, H., Lotspiech, J., Megiddo, N.: Efficient traitor tracing. Technical Report Research Report RJ 10390. IBM Almaden Research Center, San Jose, CA (March 2006)
14. Johanson, M.: A rtp to http video gateway. In: WWW 2001: Proceedings of the 10th International Conference on World Wide Web, pp. 499–503. ACM, New York (2001)
15. Kirovski, D., Malvar, H.: Robust spread-spectrum audio watermarking. In: IEEE International Conference on ICASSP 2001: Proceedings of the Acoustics, Speech, and Signal Processing 2001, pp. 1345–1348. IEEE Computer Society, Washington, DC, USA (2001)
16. Laarhoven, T.: Collusion-resistant traitor tracing schemes (2011)
17. Microsoft Corporation. IIS Smooth Streaming Transport Protocol (2009)
18. Pantos, R. (ed.): HTTP Live Streaming (2009)
19. Schulzrinne, H., Casner, S.L., Frederick, R., Jacobson, V.: RTP: A transport protocol for real-time applications. IETF Request for Comments: RFC 3550 (July 2003)

20. Skoric, B., Katzenbeisser, S., Celik, M.U.: Symmetric tardos fingerprinting codes for arbitrary alphabet sizes. Des. Codes Cryptography 46(2), 137–166 (2008)
21. Tardos, G.: Optimal probabilistic fingerprint codes. J. ACM 55(2) (2008)
22. Towes, K., Green, T.: Video content protection measures enabled by Adobe Flash Media Interactive Server 3.5 (2010)
23. White, M.A.G., Wajs, A.A.: Method and system to uniquely associate multicast content with each of multiple recipients. U.S. Patent #7058809 (June 2006)

Conjunctive Wildcard Search over Encrypted Data

Christoph Bösch, Richard Brinkman, Pieter Hartel, and Willem Jonker

University of Twente,
Enschede, The Netherlands

Abstract. Searchable encryption allows a party to search over encrypted data without decrypting it. Prior schemes in the symmetric setting deal only with exact or similar keyword matches. We describe a scheme for the problem of wildcard searches over encrypted data to make search queries more flexible, provide a security proof for our scheme and compare the computational, communication and space complexity with existing schemes. We develop an efficient scheme, using pseudorandom functions and Bloom filters, that supports wildcard searches over encrypted data. The scheme also supports conjunctive wildcard searches, efficient and secure updates and is more efficient than previous solutions. Besides, our construction is independent of the encryption method of the remote data and is practical to use in real world applications.

Keywords: Searchable Encryption, Bloom filter, Wildcard.

1 Introduction

Nowadays, remote storage is ubiquitous and widely used for services like outsourcing data to reduce operational costs or private backups. To securely store outsourced data on an untrusted server, the data should be encrypted which makes it impossible for inside and outside attackers to access the data, but at the same time the data owner loses all searching capabilities. It is desirable to support (full) searching functionality on the server side, without decrypting the data, and thus, without any loss of data confidentiality. This is typically called *searchable encryption* (SE).

Over the last decade there has been active research in the symmetric [16,10,7,8] and the public key setting [4,6]. To construct efficient schemes we focus on searchable symmetric encryption (SSE), where the same client stores and retrieves encrypted documents. Prior SSE schemes support only exact keyword matches or similarity searches, where keyword similarity is measured in the Hamming or edit distance. To get more flexibility in the search queries, we create a new construction that supports wildcard searches over encrypted data, where a wildcard may represent any number of characters. We present a construction that supports conjunctive wildcard searches, where a conjunction is the union of any number of keywords.

© Springer-Verlag Berlin Heidelberg 2011

Protocol. We consider a user $\mathcal{U}$ who stores a set of encrypted documents on an honest-but-curious [11] database server $\mathcal{S}$ that can be trusted to adhere to the protocol, but which tries to learn as much information as possible. $\mathcal{U}$ later wants to retrieve some of the documents containing a specific keyword. To do so, $\mathcal{U}$ first generates an index over his documents and then stores the index and the encrypted documents on the server. The index allows $\mathcal{U}$ to search the encrypted documents. To search for a specific keyword in the document collection, $\mathcal{U}$ creates a trapdoor for that keyword and sends this trapdoor to the server which then returns the result indicating which documents match the query and which not. $\mathcal{U}$ then decides which of the documents she wants to retrieve and sends the document ids to $\mathcal{S}$. The server returns the requested documents.

Related Work. Searchable encryption can be achieved by using the works of Ostrovsky and Goldreich [15,14,12] on *oblivious RAMs* from 1990, which hide all information including the access pattern, from a remote server. Unfortunately the scheme is not efficient in practice. The scheme needs a logarithmic number of rounds of interaction for each read and write.

The first practical scheme for searching in encrypted data in the symmetric setting was proposed by Song et al. [16] in 2000. They use a special two-layered encryption construct which is known as a sequential scan. Unfortunately, the scheme is not secure against statistical analysis across multiple queries and can leak the positions of the queried keywords in a document. The scheme has to use fix-sized words and the complexity of the encryption and search is linear in the number of words. Also it is not compatible with existing file encryption standards and has to use their specific encryption method which can be used only for plaintext data and not for example on compressed data.

Some of the above problems are addressed by Goh [10] by introducing a Bloom filter index to each document. The index makes the scheme independent of the document encryption. Goh also introduced the formal indistinguishability against chosen keyword attack (IND-CKA) and a slightly stronger IND-CKA2 adversary model.

Chang and Mitzenmacher [7] developed two index schemes, similar to Goh [10], using pre-build dictionaries. Their search schemes are independent of the encryption method and use one index per document.

Curtmola et al. [8] propose new adversarial models for searchable encryption: a non-adaptive and an adaptive one. They construct two schemes which are provably secure in these new models. The first scheme (SSE-I) is only secure against non-adaptive adversaries, but more efficient than the second scheme (SSE-II), which is also secure against adaptive adversaries.

Our Contribution. In this paper we present the first conjunctive wildcard search scheme in the symmetric setting. The scheme is proven secure against adaptive adversaries.

Structure. The rest of the paper is organized as follows. Section 2 describes the building blocks necessary for our constructions. We summarize the security definitions from Curtmola et al. [8] in Section 3. For sake of simplicity we explain

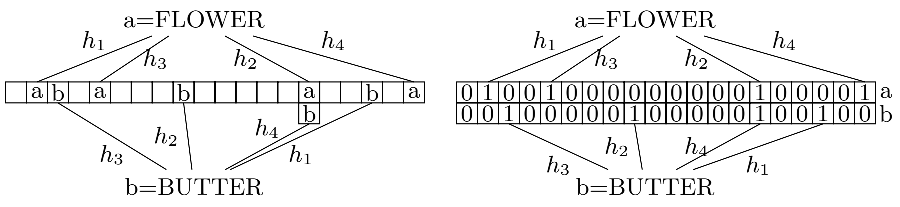

a=FLOWER
h_1
h_3
h_2
h_4
a
b
b=BUTTER
(a) normal
a=FLOWER
0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 a
0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 b
b=BUTTER
(b) alternative

Fig. 1. Bloom filter with storage

an easy basic search scheme in Section 4.1, before we describe our new masked index scheme in Section 4.2. We then present the wildcard add-on for our search scheme in Section 5. Section 6 analyses the security of our masked scheme and in Section 7 we take a look at the efficiency of the constructions. We conclude the paper in Section 8.

2 Preliminaries

Bloom Filters. A Bloom filter (BF) [3] is a data structure which is used to answer set membership queries. It is represented as an array of b bits which are initially set to 0. In general the filter uses r independent hash functions h_t, where $h_t : \{0,1\}^* \rightarrow [1,b]$ for $t \in [1,r]$, each of which maps a set element to one of the b array positions. For each element e in the set $\mathcal{S} = \{e_1, \ldots e_m\}$ the bits at positions $h_1(e), \ldots, h_r(e)$ are set to 1. To check whether an element x belongs to the set $\mathcal{S}$, we check if the bits at positions $h_1(x), \ldots, h_r(x)$ are set to 1. If so, x is considered a member of set $\mathcal{S}$. Bloom filters have a possibility of false positives, because the positions of an element may have been set by one or more other elements. With appropriate parameters the false positive probability can be reduced to a desired error rate.

Instead of r different hash functions we use a single HMAC-SHA1 [2, 13] function with r different and independent keys to create a trapdoor. This allows only legitimate users in possession of the keys to construct the correct Bloom filter and thus to add and search documents on a server.

In our constructions we use a Bloom filter with storage, as introduced by Boneh et al. [5]. Figure 1 shows two different versions of a Bloom filter with storage. Our constructions use the type (b).

Pseudorandom Generators. A pseudorandom bit generator $g : \{0,1\}^\alpha \rightarrow \{0,1\}^\beta$ is a deterministic algorithm which, given a seed of length α, outputs a binary sequence of length $\beta \gg \alpha$ that is computationally indistinguishable from a random string.

Notation. Throughout this paper we use the following notation. Let $\mathcal{D}$ be a document collection $\mathcal{D} = \{d_{id_1}, \ldots, d_{id_n}\}$, consisting of n documents. The size of a document d_{id} is denoted $|d_{id}|$, where id is a unique document identifier. Each document d_{id} consists of a set of words W_{id}. Let $\Delta_{id} = u(d_{id})$ be a dictionary of distinct words in a document d_{id}. The function $u(\cdot)$ extracts the unique words of a document. The number of distinct words per document is denoted by $|\Delta_{id}|$. We refer to $\mathcal{D}(w)$ as all the document ids containing word w and the sequence $\mathcal{D}(w_1), \ldots, \mathcal{D}(w_c)$ as the access pattern of a client.

3 Definitions

Index Scheme. Our index schemes consist of the following four algorithms:

Keygen(s): Given a security parameter s, Keygen outputs the master private key $\mathcal{K}$. This algorithm is run by the client.

BuildIndex($\mathcal{K}, \mathcal{D}$): Given the master key $\mathcal{K}$ and a document collection $\mathcal{D}$, the algorithm outputs an index $\mathcal{I}$. This algorithm is run by the client.

Trapdoor($\mathcal{K}, w$): Given the key $\mathcal{K}$ and a keyword w, Trapdoor outputs the trapdoor T_w for w. This algorithm is run by the client.

SearchIndex($T_w, \mathcal{I}$): Given a trapdoor T_w for word w and the index $\mathcal{I}$, the algorithm outputs a bit string which indicates the matched documents. This algorithm is run by the server.

3.1 Security Definitions

We use the security definitions for searchable symmetric encryption (SSE) from Curtmola et al. [8] which we summarize in this section. For detailed information we refer to the original paper [8].

Before stating the security definition for semantic security for SSE, we introduce three auxiliary notions: the *history*, the *view* and the *trace*.

History. The *history* defines the user input to the scheme. It is an interaction between the client and the server, which is determined by a document collection and a set of words that the client wishes to search for (and that we wish to hide from the adversary).

Definition 1. (History). *A* history H_u, *is an interaction between a client and a server over u queries, consisting of a document collection $\mathcal{D}$ and the keywords w_i used for u consecutive search queries. The partial history H_u^τ of a given history $H_u = (\mathcal{D}, w_1, \ldots, w_u)$, is the sequence $H_u^\tau = (\mathcal{D}, w_1, \ldots, w_\tau)$, where $\tau \leq u$.*

View. The server's *view* consists of all the information the server can gather during a protocol run. In particular, the view will consist of the index (of the document collection) and the trapdoors (of the queried words). It will also contain some additional common information, such as the number of documents in the collection and their ciphertexts. However the view should not reveal any information about the history besides the outcome and the pattern of the searches. Let $\mathcal{I}$ be the index of a document collection generated under key K, and T_{w_i}, $1 \leq i \leq u$, be the trapdoors for the words w_i queried in H_u.

Definition 2. (View). *Let $\mathcal{D}$ be a collection of n documents and $H_u = (\mathcal{D}, w_1, \ldots, w_u)$ be a history over u queries. An adversary's* view *of H_u under secret key K is defined as*

$$V_K(H_u) = (id_1, \ldots, id_n, E(d_{id_1}), \ldots, E(d_{id_n}), \mathcal{I}, T_{w_1}, \ldots, T_{w_u}) .$$

The partial view $V_K^\tau(H_u)$ of a history H_u under secret key K is the sequence

$$V_K^\tau(H_u) = (id_1, \ldots, id_n, E(d_{id_1}), \ldots, E(d_{id_n}), \mathcal{I}, T_{w_1}, \ldots, T_{w_\tau}) ,$$

where $\tau \leq u$.

Note that K refers only to the secret key for the SSE scheme and not to the encryption key of the documents.

Trace. The *trace* consists of exactly the information we are willing to leak or that the server is allowed to learn. This information includes the identifiers of the documents that contain each query word in the history and information that describes which trapdoors in the view correspond to the same underlying words in the history. The encrypted documents are also stored on the server, so the document sizes and identifiers will be leaked. We add also the sequence $(\mathcal{D}(w_1), \ldots, \mathcal{D}(w_n))$ which denotes the *access pattern* of a client and the *search pattern Π_u* of a client as any information that can be derived from knowing whether two arbitrary searches were performed for the same word or not to the trace. More formally, Π_u can be thought of as a symmetric binary matrix where $\Pi_u[i, x] = 1$ if $w_i = w_x$, and 0 otherwise, for $1 \leq i, x \leq u$.

Definition 3. (Trace). *Let $\mathcal{D}$ be a collection of n documents and $H_u = (\mathcal{D}, w_1, \ldots, w_u)$ be a history over u queries. The* trace *of H_u is the sequence*

$$Tr(H_u) = (id_1, \ldots, id_n, |d_{id_1}|, \ldots, |d_{id_n}|, \mathcal{D}(w_1), \ldots, \mathcal{D}(w_u), \Pi_u) .$$

Semantic Security. We now present the simulation-based definition for semantic security from Curtmola et al. [8]. We assume that the client initially stores a number of documents and afterwards performs an arbitrary number of search queries. For all queries $0 \leq \tau \leq u$, we require the simulator, given only a partial trace of the history, to simulate the adversary on a partial view of the same history.

Definition 4. (Adaptive Semantic Security for SSE). *An SSE scheme is adaptively semantically secure if for all $u \in \mathbb{N}$ and for all (non-uniform) probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a (non-uniform) probabilistic polynomial-time algorithm (the simulator) $\mathcal{S}$ such that for all traces Tr_u of length u, all polynomially samplable distributions $\mathcal{H}_u$ over $\{H_u : Tr(H_u) = Tr_u\}$ (i.e., the set of histories with trace Tr_u), all functions $f : \{0,1\}^m \rightarrow \{0,1\}^{v(m)}$ (where $m = |H_u|$ and $v(m) = \mathsf{poly}(m)$), all $0 \leq y \leq u$ and all polynomials p and sufficiently large s:*

$$|Pr\left[\mathcal{A}\left(V_K^\tau(H_u)\right) = f(H_u^\tau)\right] - Pr\left[\mathcal{S}\left(Tr(H_u^\tau)\right) = f(H_u^\tau)\right]| < \frac{1}{p(s)},$$

where $H_u \overset{R}{\leftarrow} \mathcal{H}_u, K \leftarrow \mathsf{Keygen}(s)$*, and the probabilities are taken over* $\mathcal{H}_u$ *and the internal coins of* Keygen*,* $\mathcal{A}$*,* $\mathcal{S}$ *and the underlying* BuildIndex *algorithm.*

4 Constructions

In this section we describe two index based constructions similar to Goh [10]. We first introduce the basic construction, which stores a keyed Bloom filter index on an untrusted server. The second construction stores a masked index on the server side. We refer to our basic search scheme as (B) and to our masked index search scheme as (M). Both constructions use a Bloom filter per document. The index can thus be represented as an $n \times b$ binary matrix where n is the number of documents and b the size of a single Bloom filter in bits. We use the words index and matrix interchangeably.

4.1 The Basic Index Scheme

To create a searchable index, we use one Bloom filter per document. We insert all distinct words of a document d_{id} in its Bloom filter BF_{id} by applying the HMAC-SHA1 function r times with r independent keys on each distinct word. All the BFs and the encrypted documents are then stored on the server.

To search in the database a trapdoor is required. This trapdoor for finding a specific keyword w in the database is derived by applying r times HMAC-SHA1 on the keyword to search for. The outcome of each HMAC-SHA1 denotes a specific position in a Bloom filter. After receiving the trapdoor, the server looks up the columns of the index specified in the trapdoor, handles them as bit strings and computes the bitwise AND on the columns. The resulting bit string indicates a match with a 1 and a non-match with a 0.

Our construction consists of the following four algorithms:

$\mathsf{Keygen}(s)$: Given a security parameter s, generate a secret master key $K = \{k_1, \ldots, k_r\}$, consisting of r independent secret keys.

$\mathsf{Trapdoor}(K, w)$: Given the key $K = \{k_1, \ldots, k_r\}$ and a word w, calculate the positions $p_t = h_{k_t}(w)$ for $t \in [1, r]$ in a Bloom filter and output the trapdoor $T_w = \{p_1, \ldots, p_r\}$, where $p_t \in [1, b]$.

$\mathsf{BuildIndex}(K, \mathcal{D})$: The input is the master secret key K and a document collection $\mathcal{D}$ comprising of a set of n documents.

1. For each $id \in [1, n]$, create the list of unique words $\Delta_{id} = u(d_{id})$ and compute for each word $w_i \in \Delta_{id}$:
   (a) the trapdoor: $T_{w_i} = \{p_1, \ldots, p_r\}$
   (b) and set the bits at the positions T_{w_i} in BF_{id} to 1.
2. Output the index $\mathcal{I} = (\mathrm{BF}_1, \ldots, \mathrm{BF}_n)^T$.

We define $\mathcal{I}[p]$ as the column vector $[\mathrm{BF}_{id}[p]]_{id \in [1,n]}$ of the matrix $\mathcal{I}$.

$\mathsf{SearchIndex}(T_w, \mathcal{I})$: Given the trapdoor $T_w = \{p_1, \ldots, p_r\}$ for word w and the index $\mathcal{I}$, take the set of columns $\{\mathcal{I}[p_t]\}_{t\in[1,r]}$ of the matrix $\mathcal{I}$. Consider each column as a bit string and output the bitwise AND of the columns.

It is easy to see, that this construction is vulnerable to correlation attacks which leak the similarity of documents upfront. This is because each word is represented by the same r positions in all Bloom filters. Another disadvantage of Bloom filters is the fact, that the number of 1's is dependent on the number of entries, in this case the number of distinct keywords per document. As a consequence, the scheme gives a good guess on the number of keywords in each document. To conceal this information we can use padding, where we add random strings to a documents distinct word list, so that the number of entries per BF is equal. To gain a higher level of security we mask the index before it is stored on an untrusted server as seen in the next section.

4.2 The Masked Index Scheme

To mask the index we use a pseudorandom generator $g(K_G, p, id)$ which takes a secret generator key K_G and the exact position of the bit to mask in the matrix (p, id) as input.

Our construction consists of the following four algorithms:

$\mathsf{Keygen}(s)$: Given a security parameter s, generate a secret master key $K = \langle K_H, K_G\rangle$, with $K_H = \{k_t\}_{t\in[1,r]}$ being r independent keys to compute the HMAC and $K_G \in \{0,1\}^*$ the key for the pseudorandom generator.

$\mathsf{Trapdoor}(K_H, w)$: Given the key $K_H = \{k_1, \ldots, k_r\}$ and a word w, output the trapdoor $T_w = \{p_1, \ldots, p_r\}$, where $p \in [1, b]$.

$\mathsf{BuildIndex}(K, \mathcal{D})$: The input is the master secret key K and a document collection $\mathcal{D}$ comprising of a set of n documents.

1. For each $id \in [1, n]$, create the list of unique words $\Delta_{id} = u(d_{id})$ and compute for each word $w_i \in \Delta_{id}$:
   (a) the trapdoor: $T_{w_i} = \{p_1, \ldots, p_r\}$
   (b) and set the bits at the positions T_{w_i} in BF_{id} to 1.
2. Create the index $\mathcal{I} = (\mathrm{BF}_1, \ldots, \mathrm{BF}_n)^T$.
3. For each position $\mathrm{BF}_{id}[p]$, with $p \in [1, b]$, compute $g(K_G, p, id)$ and create the masked index

$$\mathcal{M}[p][id] = \{\mathcal{I}[p][id] \oplus g(K_G, p, id)\}\,.$$

4. Output the masked index $\mathcal{M}$.

$\mathsf{SearchIndex}(T_w, \mathcal{M})$: Given the trapdoor $T_w = \{p_1, \ldots, p_r\}$ for word w and the masked index $\mathcal{M}$, send the set of columns $\{\mathcal{M}[p_t]\}_{t\in[1,r]}$ of the matrix $\mathcal{M}$ to the client. The client computes the set of decrypted columns

$$\mathcal{I}[p_t][id] = \{\mathcal{M}[p_t][id] \oplus g(K_G, p_t, id)\}_{t\in[1,r]}\,,$$

and outputs the bitwise AND of the columns. The resulting bit vector indicates the matched documents.

This interactive construction allows a user to decide which documents from the list of matched documents she wants to retrieve. By having a two-round protocol the scheme becomes more flexible. This is comparable with an internet search, where the search engine gives a list of results and the user can decide, which sites to download/visit. Most of the times not all of the matching documents are interesting for a user. By downloading only the desired documents, instead of all the matched documents, we do not produce unnecessary traffic. This is important for mobile users with limited bandwidth or expensive data usage fees.

4.3 Properties

Boolean Queries. As an additional feature our scheme supports conjunctive search queries, which means a boolean AND combination of two or more keywords. This is done by sending the union of several trapdoors to the server, which then sends back the result associated to those keywords. The resulting bit string indicates the documents including all of the searched words.

Secure Updates. Our search scheme supports efficient and secure updates on a document collection $\mathcal{D}$, in the sense that the client is able to Add and Delete documents from the database. A document is added by simply running the $\mathsf{BuildIndex}$ algorithms with the new documents as input. The resulting index can then be appended to the existing index stored on a server. To delete a document in our constructions, the document and the corresponding row of the index can be deleted from the server.

Adding a document can be done with the following algorithm:

$\mathsf{Add}(K, \mathcal{D})$: This Algorithm is equal to $\mathsf{BuildIndex}$. The resulting index is appended to the existing index.

Deleting a document in the unmasked index scheme can be done with the following algorithm:

$\mathsf{Delete}(id)$: Given a document id, delete d_{id} and BF_{id} from the server.

Complexity. The complexity of an update operation (add, delete) depends on the number of documents processed. The communication overhead is $O(n)$, where n is the number of documents $\mathcal{U}$ wants to add. Per document $\mathcal{U}$ has to transfer the Bloom filter of size b bit to the server. To delete a document in our schemes, $\mathcal{U}$ can simply delete the document and the corresponding row from the index by sending the id to the server.

Security. During an update operation, $\mathcal{U}$ reveals only the number of documents processed. The newly added BFs look like random strings.

5 Wildcard Add-On

In this section we introduce a simple wildcard add-on that can be used with most search schemes. The main idea behind our wildcard search is to pre-process the words that will be inserted into the index: for each distinct word we create all the wildcardified variants of the word as shown in Algorithm 1. The individual characters of a word are denoted by $w_i[j]_{j \in [1,\lambda]}$ where $w_i[x : y]$ denotes the characters x to y.

For example, the keyword `flower` will be represented in a single wildcard scheme as {`flower, *flower, flower*, *lower, ..., flowe*, *ower, ..., flow*, *wer, f*er, fl*r, flo*, *er, f*r, fl*, *r, f*`}. Thus all possible variations of the word are created.

The number of all these single wildcard combinations per distinct word is computed by

$$\left(\sum_{j=1}^{\lambda} j\right) + 2 = \frac{\lambda(\lambda+1)}{2} + 2,$$

where λ is the length of the word w_i.

Algorithm 1. Algorithm for Wildcardifying Words

```
Input: A word $w_i$
Output: $\Omega_i$: all wildcardified versions of the word $w_i$
1: wild_carded_words = $[w_i, *w_i, w_i*]$
2: for wild_card_size = 1 to wordLength do
3:     for $v = 1$ to (wordLength - wild_card_size +1) do
4:         wild_carded_words.append($w_i[1 : v] + * + w_i[v + \text{wild\_card\_size} : \lambda]$)
5:     end for
6: end for
```

For our wildcard search scheme we insert not only the keywords, but all the wildcardified versions of a word into the index. Hence the scheme transforms the problem of a wildcard search into a lookup for an exact match. The scheme still supports conjunctive search queries.

Example 1. A search for the word chin* will return all the document ids containing a word starting with chin*, like china, chinatown, chinaware, chinchilla, chine, chinese, chinked, chinless, ...

Multiple Wildcards. If desired, it is possible to add multiple wildcards at the cost of more pre-processing and server storage space. Thus it is possible, to add the support for two or more wildcards (e.g., `*owe*` or `f*o*r`).

6 Security Proof

We now provide the security proof for our masked index scheme. At this point we do not take updates and conjunctive queries into account. HMAC-SHA1 is used as the hash function for the Bloom filter. Bellare [1] proved, that HMAC is a pseudorandom function.

Theorem 1. *If h and g are secure pseudorandom functions, our masked search scheme described in Section 4.2 including the wildcard add-on explained in Section 5 is an adaptively secure SSE scheme.*

Proof. Let $u \in \mathbb{N}$, and let $\mathcal{A}$ be a probabilistic polynomial-time adversary. We describe a probabilistic polynomial-time simulator $\mathcal{S}$ such that for all polynomially-bounded functions f and all distributions $\mathcal{H}_u$, $\mathcal{S}$ can simulate the partial view of an adversary $\mathcal{A}(V_K^\tau(H_u))$ given only the trace of a partial history $Tr(H_u^\tau)$ for all $0 \le \tau \le u$ with probability negligibly close to 1. For all $0 \le \tau \le u$, we show that $\mathcal{S}(Tr(H_u^\tau))$ can generate a simulated view $\mathfrak{V}_u^\tau$ that is indistinguishable from $V_K^\tau(H_u)$. Let

$$Tr(H_u^\tau) = (id_1, \ldots, id_n, |d_{id_1}|, \ldots, |d_{id_n}|, b, \mathcal{D}(w_1), \ldots, \mathcal{D}(w_\tau), \Pi_\tau).$$

be the trace of an execution after τ search queries and let H_u be a history consisting of u search queries such that $Tr(H_u) = Tr_u$. The simulator $\mathcal{S}$ works as follows:

With the information from the trace, $\mathcal{S}$ chooses n random values $R_1, \ldots, R_n$ such that $|R_i| = |d_i|$ for all $i = 1, \ldots, n$. $\mathcal{S}$ also includes the document identifiers, known from the trace, in the partial view. Then the simulator $\mathcal{S}$ generates a simulated index $\mathfrak{M} = (B_1, \ldots, B_n)^T$ with random $B_i \in \{0,1\}^b$, for $i \in [1,n]$. $\mathfrak{M}$ will be included in all partial views $\mathfrak{V}_u^\tau$ used to simulate $\mathcal{A}$. Next $\mathcal{S}$ simulates the trapdoor for query $\tau, (1 \le \tau \le u)$ in sequence. If $\Pi_\tau[j,\tau] = 1$ for some $1 \le j < \tau$ set $\mathfrak{T}_\tau = \mathfrak{T}_j$. Otherwise $\mathcal{S}$ picks a random value rnd, calculates $p_t = h_{k_t}(rnd)$ for $t \in [1,r]$ and sets $\mathfrak{T}_\tau = \{p_1, \ldots, p_r\}$, such that for $1 \le j < \tau, \mathfrak{T}_\tau \ne \mathfrak{T}_j$. Then $\mathcal{S}$ constructs for all τ a simulated view

$$\mathfrak{V}_u^\tau = (id_1, \ldots, id_n, E(R_1), \ldots, E(R_n), \mathfrak{M}, \mathfrak{T}_1, \ldots, \mathfrak{T}_\tau),$$

and eventually outputs $\mathcal{A}(\mathfrak{V}_u^\tau)$. We now claim that $\mathfrak{V}_u^\tau$ is indistinguishable from

$$V_K^\tau(H_u) = (id_1, \ldots, id_n, E(d_{id_1}), \ldots, E(d_{id_n}), \mathcal{M}, T_1, \ldots, T_\tau).$$

Therefore we state that for all $i \in [1,n]$, id_i in $V_K^0(H_u)$ and $\mathfrak{V}_u^0$ are identical and thus indistinguishable. Also, $E(\cdot)$ is a semantically secure encryption algorithm, thus $E(d_i)$ is indistinguishable from $E(R_i)$ of the same length. Given the $\mathsf{BuildIndex}$ algorithm, it is clear that $\mathfrak{M}$ is indistinguishable from $\mathcal{M}$. Otherwise one could distinguish between a random string B of size b and $[BF[p] \oplus g(K_G, p)]_{p \in [1,b]}$, the bitwise XOR of a Bloom filter of size b and the output of the pseudorandom generator $g(\cdot)$. It is easy to see that the trapdoors are indistinguishable, otherwise one could distinguish $h_k(\Omega)$ from $h_k(rnd)$. Thus, $\mathfrak{V}_u^\tau$ is indistinguishable from $V_K^\tau(H_u)$, for all $0 \le \tau \le u$. □

Updates. In our scenario, intermixing queries and updates is equivalent to first update and then query. Incremental updates can be aggregated to one BuildIndex over a larger document set. Thus we can proceed with the proof of Theorem 1.

Conjunctive Queries. The above proof also holds if we search for a conjunctive set of words. Imagine that we substitute the words w_0 and w_1 with the two sets $(w_{0,1}, \ldots, w_{0,l})$ and $(w_{1,1}, \ldots, w_{1,l})$. Then proceed with the proof of Theorem 1.

Stronger Security. Note that to gain a higher level of security it is possible to split the large Bloom filter into several smaller parts and store the parts on different non communicating servers. Another way of increasing the security by not revealing the access pattern is to store the index and the documents on two different non communicating servers.

7 Performance

We now consider the efficiency of our constructions where the efficiency is measured in terms of the computation, communication and space complexity.

Computational Complexity. Table 2 shows the efficiency of our schemes compared to others in terms of computational complexity. The Trapdoor, BuildIndex and SearchIndex columns describe the computational complexity of the algorithms. The column Server gives the computation on the server side, whereas Client shows the computational complexity on the client.

The Trapdoor algorithm is a constant time operation. The BuildIndex algorithm has to process each distinct word per document. Thus the complexity is $O(n|\Delta|)$. Because the index is stored as a Bloom filter with storage (see Figure 1(b)), the SearchIndex algorithm is a simple table lookup and takes time $O(1)$. However the table lookup does not give us the result of the query. Thus after a basic index search the server has to compute a bitwise AND of the matrix columns labelled by the trapdoor, which is a $O(n)$ operation. With the masked index the server is not able to compute the result and has to send the matrix columns to the client, which is then able to decrypt the columns by an XOR operation and then performs the AND on the unmasked columns. Thus the client computation is $O(n)$. Note that in both of our schemes the server and client computations are AND and/or XOR operations and thus are efficient. Table 2 shows that in the big O notation the scheme of Curtmola et al. is more efficient than our schemes but in practice n AND operations are more efficient than $|\mathcal{D}(w)|$ encryptions.

The scheme of Song et al. (SWP) [16] does not use an index. Thus the BuildIndex field is marked with a "-". The SearchIndex algorithm denotes the search through the encrypted documents. In the SWP scheme all the words per document have to be searched and so the complexity is $O(nq)$ where n is the number of documents and the q the number of words per document.

Communication and Space Complexity. Table 3 compares the space complexity of different search schemes. Index describes the storage space for the

Table 1. Number example of communication and space complexity. Parameters: Keywords to search = 1000, average word length = 5, FP-rate = 0.01, k = 6, b = 153000.

Documents	Index	Trapdoor	Result
1000	18.24 MB	108 b	750 B
5000	91.21 MB	108 b	3.66 kB
10000	182.42 MB	108 b	7.32 kB

Table 2. Computational performance of different search schemes, where n is the number of documents in the database and q the number of words per document. The number of distinct words per document is denoted by $|\Delta|$ and $|\mathcal{D}(w)|$ denotes the number of documents containing the keyword w. The number of all distinct words in the database is denoted by $|W_{db}|$. The asterisk * denotes a bitwise AND and/or XOR. The two asterisks ** refer to the use of a so-called *FKS dictionary* introduced by Fredman et al. [9].

Scheme	Trapdoor	BuildIndex	SearchIndex	Server	Client
SWP [16]	$O(1)$	$O(nq)$	$O(nq)$	$O(nq)$	$O(\|\mathcal{D}(w)\|q)$
Goh [10]	$O(1)$	$O(n\|\Delta\|)$	$O(n)$	$O(n)$	$O(1)$
CM2 [7]	$O(\log \|W_{db}\|)$	$O(n\|\Delta\|)$	$O(n)$	$O(n)$	$O(1)$
SSE-1 [8]	$O(1)$	$O(n\|\Delta\|)$	$O(1)^{**}$	$O(\|\mathcal{D}(w)\|)$	$O(1)$
Our (M)	$O(1)$	$O(n\|\Delta\|)$	$O(1)$	$O(1)$	$O(n)^*$

Table 3. Communication and space complexity of different search schemes, where n is the number of documents in the database and $|\mathcal{D}(w)|$ the number of documents containing the keyword w. The total size of the plaintext document collection in units, where a unit is the smallest possible size for a word, is denoted by m and the number of all distinct words in the database is denoted by $|W_{db}|$.

Scheme	Index	Trapdoor	Result
SWP [16]	-	$O(1)$	$O(\|\mathcal{D}(w)\|)$
Goh [10]	$O(n)$	$O(1)$	$O(n)$
CM2 [7]	$O(n)$	$O(1)$	$O(n)$
SSE-1 [8]	$O(m) + O(\|W_{db}\|)$	$O(1)$	$O(\|\mathcal{D}(w)\|)$
Our (M)	$O(n)$	$O(1)$	$O(n)$

index on the server side. The Trapdoor column shows the size of a trapdoor and the Result column describes the size of the results that have to be transferred to the client. In both of our schemes the index can be seen as a $n \times b$-matrix, where n is the number of documents and b the size of the Bloom filter. Thus the server has to store nb bits, where b is a constant. The trapdoor has a constant size $O(1)$ and the size of the result vector is $O(n)$ because it is dependent on the number of documents in the database.

Example 2. Assume a user who wants to search for 1000 keywords. The average word length in the English language is 5 characters per word. Thus we end up with 17,000 wildcardified words. To achieve a false positive rate of 0.01, we set k = 6 and b = 153000. The resulting sizes for different document collections can be found in Table 1.

8 Conclusion

We examined the problem of wildcard searches over encrypted data in the symmetric setting and proposed a searchable encryption scheme similar to Goh [10] which supports wildcards that can be either any single character or a string of characters inside a word. The scheme also supports conjunctive search queries with any number of keywords. We proposed two variants of our scheme which differ in the security of the index and the communication overhead. The first scheme is more efficient in terms of computation and communication, while the second scheme is more secure in the sense that we leak less information about the index. Our masked scheme is proven secure against adaptive adversaries. Our schemes are more efficient than previous search schemes and are practical to use in real world applications.

References

1. Bellare, M.: New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
2. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
3. Bloom, B.H.: Space/Time Trade-Offs in Hash Coding with Allowable Errors. Commun. ACM 13(7), 422–426 (1970)
4. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
5. Boneh, D., Kushilevitz, E., Ostrovsky, R., Skeith III, W.E.: Public Key Encryption that Allows PIR Queries. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 50–67. Springer, Heidelberg (2007)
6. Boneh, D., Waters, B.: Conjunctive, Subset, and Range Queries on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
7. Chang, Y.-C., Mitzenmacher, M.: Privacy Preserving Keyword Searches on Remote Encrypted Data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
8. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In: CCS 2006: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 79–88. ACM, New York (2006)

9. Fredman, M.L., Komlós, J., Szemerédi, E.: Storing a Sparse Table with 0(1) Worst Case Access Time. J. ACM 31(3), 538–544 (1984)
10. Goh, E.-J.: Secure Indexes. Cryptology ePrint Archive, Report 2003/216 (2003)
11. Goldreich, O.: Secure Multi-Party Computation. Working draft (October 2002)
12. Goldreich, O., Ostrovsky, R.: Software Protection and Simulation on Oblivious RAMs. J. ACM 43(3), 431–473 (1996)
13. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC2104, Internet Engineering Task Force (IETF) (February 1997)
14. Ostrovsky, R.: Efficient Computation on Oblivious RAMs. In: Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, Baltimore, Maryland, USA, May 14-16, pp. 514–523. ACM, New York (1990)
15. Ostrovsky, R.: Software Protection and Simulations on Oblivious RAMs. PhD thesis. MIT (1992)
16. Song, D.X., Wagner, D., Perrig, A.: Practical Techniques for Searches on Encrypted Data. In: SP 2000: Proceedings of the 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society, Washington, DC, USA (2000)

Adjusting the Trade-Off between Privacy Guarantees and Computational Cost in Secure Hardware PIR

Spiridon Bakiras[1] and Konstantinos F. Nikolopoulos[2]

[1] John Jay College, City University of New York
sbakiras@jjay.cuny.edu
[2] The Graduate Center, City University of New York
knikolopoulos@gc.cuny.edu

Abstract. Database queries present a potential privacy risk to users, as they may disclose sensitive information about the person issuing the query. Consequently, privacy preserving query processing has gained significant attention in the literature, and numerous techniques have been proposed that seek to hide the content of the queries from the database server. Secure hardware-assisted private information retrieval (PIR) is currently the only practical solution that can be leveraged to build algorithms that provide perfect privacy. Nevertheless, existing approaches feature *amortized* page retrieval costs and, for large databases, some queries may lead to excessive delays, essentially taking the database server offline for large periods of time. In this paper, we address this drawback and introduce a novel approach that sacrifices some degree of privacy in order to provide *fast* and *constant* query response times. Our method leverages the internal cache of the secure hardware to constantly reshuffle the database pages in order to create sufficient uncertainty regarding the exact location of an arbitrary page. We give a formal definition of the privacy level of our algorithm and illustrate how to enforce it in practice. Based on the performance characteristics of the current state-of-the-art secure hardware platforms, we show that our method can provide low page access times, even for very large databases.

1 Introduction

Internet users are becoming increasingly wary of the potential privacy risks associated with their everyday online activities. Web search engines, for example, maintain detailed logs of every query that they receive. However, with sophisticated data mining techniques, these query logs can reveal sensitive information about a user's lifestyle, health, habits, etc. [4,15]. Similarly, the emergence of location based services (LBS) allows mobile users to browse points of interest (e.g., restaurants) in their surroundings. Since these queries are also logged at the LBS provider, a user's location over a period of time can be tracked with very high accuracy [23].

Clearly, ordinary database queries involve an inherent privacy risk for users and, as a result, privacy preserving query processing is an emerging research

© Springer-Verlag Berlin Heidelberg 2011

field in the database community. A popular approach that enhances the level of privacy in certain applications, is anonymity. The central principle of anonymity is to inject sufficient noise into a query, so that the user has *plausible deniability* over the exact content of the query. For instance, the client could combine the real query with several dummy ones (that are typically unrelated) or alter slightly the query parameters. Algorithms based on anonymity have been proposed for both text search engines (e.g., [21,22]) and location based services (e.g., [3,8,16,20]). However, since the database server has access to the plaintext queries, it may be able to determine the real content of a query using background knowledge (e.g., detailed information about a specific user).

Data encryption is another technique that can be leveraged to hide the content of a query. In this scenario, the server interacts with an encrypted version of the original database. Queries are also encrypted in a similar fashion and, thus, the server can not deduce any information about the query content. Research work is this area has focused on developing efficient encryption algorithms that facilitate exact query processing at the server side [1,2]. The limitation of encryption schemes, however, is that two identical queries always produce the same encrypted result. Consequently, if the server has knowledge of the access patterns of the database records (i.e., their relative popularities), it can extract some information about a query through the records included in the result set.

Private information retrieval (PIR) is the only solution available that can be leveraged to build algorithms that provide perfect privacy. In particular, PIR protocols [7] allow a client to retrieve any record from a database, while making it impossible for a computationally bounded server to determine which record was retrieved. Note that, when PIR is employed, the server cannot perform the actual query processing. Instead, the client accesses (privately) the disk-resident index structure at the database server and resolves the query locally through a series of PIR retrievals [23]. Currently, secure hardware-assisted PIR is the only *practical* PIR construction. It is implemented on top of a tamper-resistant CPU (secure hardware), which acts as a proxy between the server and the clients. Nevertheless, existing approaches feature *amortized* page retrieval costs, because they necessitate periodic reshuffle operations on the database. As a result, some queries may lead to excessive delays, essentially taking the database server offline for large periods of time.

In this paper, we address this drawback and introduce a novel approach that sacrifices some degree of privacy in order to provide *fast* and *constant* query response times. The goal is to design a system that balances the trade-off between computational cost and privacy guarantees. In other words, we aim to provide a much stronger notion of privacy compared to anonymity or encryption based schemes, but with a computational cost that is considerably lower compared to existing PIR techniques. Such a system would benefit applications that do not require perfect privacy, but are instead satisfied with a sufficient level of uncertainty.

Our algorithm initially encrypts and obliviously permutes the database pages. Each page is then retrieved efficiently by accessing (through the secure hardware) its encrypted version from the server's disk. To further enhance the privacy of our approach, we introduce a randomized algorithm that constantly reshuffles the underlying pages in order to create sufficient uncertainty regarding the exact location of an arbitrary page. The algorithm works by randomly moving every requested page to a new location on the disk. In particular, it leverages a built-in cache at the secure hardware that stores a fixed number of previously retrieved pages. Reshuffling occurs during each page request, with a random page from the cache being written to a new location on the disk. We give a formal definition of the privacy level of this approach and illustrate how to enforce it in practice. Based on the performance characteristics of the current state-of-the-art secure hardware platforms, we show that our method can provide low page access times, even for very large databases. In summary, the contributions of our work are the following.

– We propose a novel architecture, based on state-of-the-art secure hardware, that reduces significantly the cost of private page retrievals compared to existing PIR based techniques.
– We formally define the privacy level of our approach and use analytical models to derive the corresponding security parameter.
– We evaluate the performance of our method, using (i) analytical results from a secure hardware deployment and (ii) measurements from a software implementation. We show that, given sufficient secure storage capacity, our system can achieve sub-second query response times, even for TB-sized databases.

The remainder of this paper is organized as follows. Section 2 reviews previous work on database privacy and private information retrieval techniques. Section 3 describes the architecture of our approach and outlines the underlying adversarial model. Section 4 introduces our private page retrieval algorithm and Section 5 presents the analytical results from a secure hardware implementation. Finally, Section 6 concludes the paper.

2 Related Work

PIR was first introduced by Chor et al. [7], and is formally defined as follows. The server holds a database, which is assumed to be a binary string X of length n. The client wants to retrieve the i-th bit (x_i) of the database, without the server knowing the value of the index i. In general, PIR protocols can be classified into three main categories: information theoretic, computational, and secure hardware.

First, *information theoretic* PIR [5,7,12,27] ensures that the query discloses no information about the retrieved bit, even if the server has unbounded computational power. However, these protocols are not practical, as they require that the database be replicated into k *non-colluding* servers. On the other hand, *computational* PIR protocols [6,10,18,19] work with a single server, and employ well known cryptographic primitives that guarantee query privacy for a computationally bounded server. Nevertheless, these protocols are extremely expensive for large databases, as they require at least one modular multiplication for every bit of the database.

Finally, *secure hardware* PIR [14,24,25,26] relies on a tamper resistant CPU (located at the server side), which acts as a proxy between the clients and the server. These protocols are significantly faster than computational PIR, because they do not need to scan the whole database for every query. Wang et al. [24] utilize the internal storage of the secure hardware that can hold k out of n database pages. Every request inserts a new page into the secure storage and, when the storage capacity is reached, the database is reshuffled. Therefore, the amortized computational cost of this approach is $O(n/k)$. Ref. [14,25,26] leverage the Oblivious RAM model [13], which arranges the database pages into a pyramid-like structure. To achieve access pattern privacy, (i) every level of the structure is accessed during a page retrieval and (ii) the pyramid levels are periodically reshuffled by the secure hardware. Iliev and Smith [14] propose a method with $O(\sqrt{n}\log n)$ amortized computational cost, while Williams and Sion [25] improve this amortized cost to $O(\log^2 n)$. Currently, the state-of-the-art approach is due to Williams et al. [26], and provides an amortized logarithmic computational cost of $O(\log n \log\log n)$. However, due to the periodic reshuffling of the pyramid levels, the response time of a single PIR retrieval may vary from hundreds of milliseconds to thousands of seconds (as illustrated in [26]).

PIR based solutions have been explored previously in the context of spatial nearest neighbor queries. In particular, Khoshgozaran et al. [17] and Papadopoulos et al. [23] utilize secure hardware protocols, while Ghinita et al. [11] employ an expensive computational PIR algorithm [18]. Ref. [23] is a more general and comprehensive study on the applicability of PIR protocols on multi-level index structures. The authors introduce a solution that provides perfect privacy, and also present a detailed experimental evaluation based on secure hardware [25] simulations. Their results show that query processing may require tens of seconds, even for moderate databases, due to the large number of PIR retrievals on the underlying disk-resident index structures. Motivated by this fact, we propose an alternative approach that sacrifices some degree of privacy in order to reduce significantly the query processing cost.

3 Preliminaries

Section 3.1 describes the basic architecture of our approach and Section 3.2 outlines the underlying threat model.

3.1 System Architecture

Figure 1 illustrates the proposed system architecture. The secure hardware is a *tamper resistant* CPU, such as the IBM 4764 PCI-X secure coprocessor[1]. It is attached at the server machine, but it can be trusted to operate without

[1] `http://www-03.ibm.com/security/cryptocards/pcixcc/overview.shtml`

any interference from the server. Specifically, it includes tamper detecting and responding circuitry that, in the event of an attack, destroys all the critical keys and certificates. The secure hardware incorporates an internal cache (up to 64MB for the IBM 4764 secure coprocessor) that is inaccessible by the server, and also has direct access to the server's disk.

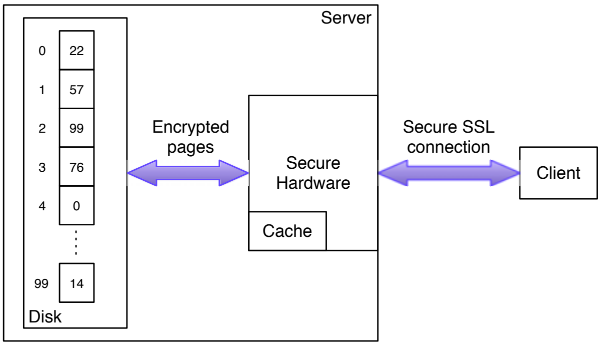


Fig. 1. System architecture

Note that the secure hardware is only necessary in the *three-party* querying model, i.e., when any client (including the adversary) is allowed to query the database server. Nevertheless, our methods are also applicable in the *two-party* querying model. The two-party model applies to the *database outsourcing* paradigm, where the data owner is the only client that accesses the database. In this setting, the owner outsources its data to a third-party service provider and wishes to access these data in a private manner. Since the data owner is the sole client in this architecture, there is no need for a secure hardware platform at the service provider. Instead, the functionality of the secure hardware can be implemented entirely at the owner's side (physically isolated from the adversary), using any standard server configuration. We explore the feasibility of this approach in Section 5.

In our problem formulation, we consider a database consisting of n pages (Table 1 summarizes the symbols used throughout the paper). Each page is a tuple $\langle id, data \rangle$, where the id attribute uniquely identifies the page. Prior to query processing, the secure hardware encrypts and obliviously permutes the database pages. It utilizes a *symmetric-key* encryption algorithm, such as AES [9], and the encryption key is secret from both the database server and the clients. Clients communicate with the secure hardware via secure SSL connections. A client query $Q(i)$ is simply a request to retrieve the page with $id = i$ from the database (we assume pages are assigned id values ranging from 0 to $n - 1$). To facilitate query processing, the secure hardware stores in its cache a look-up table that maps each page id to its actual position on the disk. After identifying the corresponding position, the secure hardware retrieves the page from the disk, decrypts it, and finally transmits it to the client via the secure connection.

Table 1. Summary of symbols

Symbol	Description
n	Database size (number of pages)
k	Block size (number of pages)
T	Number of blocks in database ($= n/k$)
m	Cache capacity (number of pages)
B	Page size (bytes)

To provide perfect query privacy, previous approaches apply periodically an oblivious permutation algorithm to reshuffle the database pages. Note that, after the reshuffling operation, every database page has an equal probability ($= 1/n$) of landing in any of the n available locations. Consequently, any query that accesses a new page from the disk becomes indistinguishable from any other query. In this work, we aim to relax this stringent constraint and allow pages to land in different disk locations according to a non-uniform distribution. Unlike prior methods, we do not reshuffle the entire database at once; instead, during each request instant, *one* previously retrieved page (that resides temporarily inside the cache) is relocated to a new position on the disk. In particular, for any value $c \geq 1$, we introduce the notion of *c-approximate PIR* as follows.

Definition 1. *A scheme provides* ***c-approximate PIR*** *if, after moving a single page p to a new location on the disk and for any pair of disk locations l_i, l_j, the probability of p landing in location l_i is at most c times larger than the probability of landing in location l_j.*

The value c is the privacy parameter of our approach, as it determines the variability of the distribution that models the individual page relocation process. Smaller values of c result in better privacy, while the case $c = 1$ offers perfect privacy (i.e., equivalent to PIR).

3.2 Adversarial Model

We assume that the adversary is the server itself, and its goal is to derive any non-trivial information regarding the *id* of a requested page. Because of the underlying secure SSL connections, both the client queries and the generated replies are unreadable by the server. Nevertheless, the server can see the accessed locations on the disk and has knowledge of all the algorithms that are implemented inside the secure hardware. We also assume that the server can only perform polynomial time computations and is "curious but not malicious" (i.e., it will not tamper will the actual data).

4 Private Page Retrieval Algorithm

Section 4.1 describes the page retrieval algorithm, while Section 4.2 provides an analytical model that quantifies its privacy level. Section 4.3 illustrates the database update procedure.

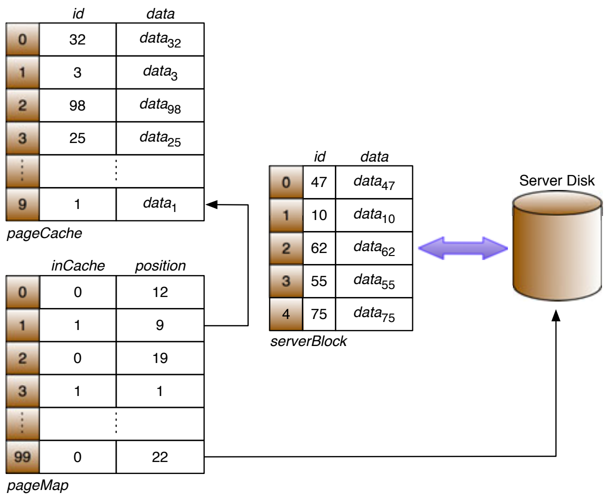


Fig. 2. Data structures at the secure hardware

4.1 Algorithm

Our approach leverages the built-in cache at the secure hardware to *obliviously mix* a pool of database pages and copy them into random positions on the disk. We assume that the cache can store a total of m pages and employs a *randomized* cache replacement policy. Note that the purpose of the cache is not to improve the page retrieval time, but to facilitate this *continuous* page reshuffling process.

During each page request, the algorithm retrieves a fixed number of $k+1$ pages, where k is the security parameter. In particular, the secure hardware initially reads (in a round-robin manner) a *block* of k contiguous pages. Specifically, on the first request it accesses the database pages at locations 0 through $k-1$, next the pages at locations k through $2k-1$, etc. The $(k+1)$-th page that is read is either the page requested by the client or a random one (the detailed algorithm is explained shortly). The reason for reading multiple pages is to guarantee that any cached page has a non-negligible probability of being written to any location on the disk (discussed in Section 4.2). If n is not a multiple of k, the secure hardware inserts an appropriate number of dummy pages during the initial reshuffling stage.

Figure 2 shows the data structures maintained at the secure hardware. First, the cache is implemented as a vector (*pageCache*) holding m database pages. *pageMap* is a vector of size n and corresponds to the look-up table for all the database pages. Each entry in *pageMap* is a tuple $\langle inCache, position\rangle$. Attribute *inCache* uses a single bit that, when set, indicates that the corresponding page is stored inside the cache. Attribute *position* is an integer value that has a dual interpretation: if $inCache = 1$, it represents the index in the *pageCache* vector where the page is stored; otherwise, it identifies the location of that page at the server disk under the current permutation order (see Figure 2). Finally, *serverBlock* is the vector (of size $k+1$) that temporarily stores the pages that

Retrieve(i)

```
    // read next block (of size $k$) in a round-robin fashion
1:  $serverBlock[0..k-1] \leftarrow read(nextBlock)$
2:  **if** $(pageMap[i].inCache$ **or** $i \in serverBlock)$
       // select a random page that is not cached
       // and is not retrieved in $serverBlock$
3:     **do**
4:        $p \leftarrow random(0, n-1)$
5:     **while** $(pageMap[p].inCache$ **or** $p \in serverBlock)$
6:     **if** $(pageMap[i].inCache)$
7:        $result \leftarrow pageCache[pageMap[i].position]$
    // else use requested page
8:  **else**
9:     $p \leftarrow i$
    // read page $p$ from the disk
10: $serverBlock[k] \leftarrow read(pageMap[p].position)$
    // decrypt all pages in $serverBlock$
11: $decrypt(serverBlock)$
12: if $(!pageMap[i].inCache)$
13:    $q \leftarrow$ index of page $i$ in $serverBlock$
14:    $result \leftarrow serverBlock[q]$
15: **else**
16:    $q \leftarrow k$
    // select a random page from the block
17:    $r \leftarrow random(0, k-1)$
18: $swap(serverBlock[r], serverBlock[q])$
    // select random page from cache
19: $s \leftarrow random(0, m-1)$
20: $swap(pageCache[s], serverBlock[r])$
    // re-encrypt all pages in $serverBlock$ (with a new nonce)
21: $encrypt(serverBlock)$
    // write updated pages at the disk
22: $write(serverBlock)$
    // update $pageMap$ (3 pages)
23: $update(pageMap[pageCache[s]])$
24: $update(pageMap[serverBlock[r]])$
25: $update(pageMap[serverBlock[q]])$
    // send page $i$ to the client over the SSL connection
26: **return** $result$
```

Fig. 3. The page retrieval algorithm

are written to or read from the disk. In the sample configuration of Figure 2, $n = 100$, $m = 10$, and $k = 4$.

The page retrieval algorithm is shown in Figure 3, and operates as follows. First, the client sends a query to the secure hardware, containing the *id* of the required page (e.g., page i). The secure hardware then reads and stores into *serverBlock* the next block of k pages, according to the round-robin schedule. Next, it accesses $pageMap[i]$ and identifies the current location of that page. If page i is located at the server and is not included in the *serverBlock* vector, the page is retrieved from the corresponding location on the disk and stored into *serverBlock*. If, on the other hand, page i is included in the *serverBlock* vector, the secure hardware selects a random page from the database that is not currently cached or stored into *serverBlock*.

Subsequently, the secure hardware decrypts all $k+1$ pages in *serverBlock* and extracts the requested page. It then selects a random page from the cache and replaces it with the newly requested page i. Similarly, the cached page is copied into *serverBlock*, overwriting page i. However, to ensure that the cached page is moved to any of the k locations in the read block (corresponding to the first k pages in *serverBlock*) with equal probability, the requested page initially swaps places with a random page in the block (line 18). Next, the pages in *serverBlock* are re-encrypted with a *new* random nonce, and are eventually transferred back to the server's disk. Finally, the secure hardware modifies the necessary entries (for the swapped pages) at the *pageMap* vector.

In the case where the requested page produces a cache hit, the secure hardware retrieves a random page p from the disk and repeats the same steps as above, i.e., as if page p was requested by the client. To summarize, during every query, the requested page (or a random page, in the case of a cache hit) is stored into the cache and a random page from the cache is moved to one of the k locations in the block that was accessed as part of that request. Note that, due to the randomized cache replacement policy, a certain cached page may be evicted while it is being requested by the client. Also, to avoid timing attacks, a cached page is not returned immediately to the client, because that would reveal the cache hit to the adversary.

4.2 Security Analysis

The page retrieval algorithm works by spreading the accesses for a single page over multiple disk locations. Once a page is requested and moves into the cache, it will be relocated to a new position during a subsequent request. Consequently, an adversary can only track probabilistically the location of an arbitrary page within the server's disk. Our goal is to properly adjust the block size k, in order to meet the privacy requirements of the c-approximate PIR definition (Section 3.1).

Consider a sequence of client requests at instants $t = 0, 1, 2, \ldots$. Assume that page p is copied into the cache during a client request at $t = 0$. Then, the probability that it moves back to the disk at time $t \geq 1$ is computed as:

$$P^t = \left(1 - \frac{1}{m}\right)^{t-1} \cdot \frac{1}{m} \tag{1}$$

Therefore, if the secure hardware accesses a set of k locations (from a single block) $\mathcal{L}_t = \{l_1, l_2, \ldots, l_k\}$ during the request at time t, the probability that page p is relocated to position l_j $(1 \leq j \leq k)$ is equal to:

$$P^t_{p \to l_j} = \left(1 - \frac{1}{m}\right)^{t-1} \cdot \frac{1}{m} \cdot \frac{1}{k} \tag{2}$$

The value k is the security parameter of our approach, since it controls the time interval $T = n/k$ (given as number of requests) that is required to scan every location on the disk exactly once through the round-robin schedule. Note that Equation (2) is a monotonically decreasing function, so the k locations that are accessed at $t = 1$ have the highest probability of hosting page p. Specifically, for the locations $l_j \in \mathcal{L}_1$, the probability that p is relocated there is:

$$P^1_{p \to l_j} = \sum_{i=0}^{\infty} \left(1 - \frac{1}{m}\right)^{T \cdot i} \cdot \frac{1}{m} \cdot \frac{1}{k} \tag{3}$$

Similarly, the locations $l_j \in \mathcal{L}_T$ have the lowest probability of storing page p:

$$P^T_{p \to l_j} = \sum_{i=0}^{\infty} \left(1 - \frac{1}{m}\right)^{(i+1) \cdot T - 1} \cdot \frac{1}{m} \cdot \frac{1}{k} \tag{4}$$

Consequently, the value of k can be determined by setting

$$\frac{P^1_{p \to l_j}}{P^T_{p \to l_j}} = \frac{1}{\left(1 - \frac{1}{m}\right)^{T-1}} = \frac{1}{\left(1 - \frac{1}{m}\right)^{\frac{n}{k}-1}} = c \tag{5}$$

Solving the above equation, we get:

$$k = \frac{n}{\frac{\log(1/c)}{\log(1-1/m)} + 1} \tag{6}$$

Note that, the value $c = 1$ corresponds to the trivial case of PIR, i.e., when the whole database is read for every request ($k = n$). On the other hand, a value such as $c = 2$ would indicate that any location is *at most* twice as likely to host a previously cached page as any other location on the disk. For a given database size n and privacy parameter c, the value of the security parameter k is determined by the available cache capacity. As evident in Equation (5), for a fixed value of T, the privacy parameter c converges towards 1 as the value of m increases.

4.3 Database Updates

A final remark concerns the handling of database updates in our system architecture. Similar to query processing, the database owner interacts only with the secure hardware through a secure SSL connection. Our system can handle trivially any type of updates, including insertions, deletions, and page modifications. In particular, every database update is treated as a regular query, i.e., the secure hardware (i) retrieves $k + 1$ pages from the disk, (ii) swaps one page from the cache with one of the retrieved pages, and (iii) writes the $k + 1$ pages back to the disk after re-encrypting them. Consequently, the type of update operation performed on the database is kept secret from the server.

Deletions are handled as cache hits, i.e., the $(k + 1)$-th page is selected randomly. Additionally, if the deleted page is stored inside the cache, it is always selected to swap positions with one of the k pages in the block. Finally, the *position* attribute of the *pageMap* entry for that page is set to a reserved value (e.g., all 1's) that signifies the deletion event. Note that, if there are numerous page deletions on the database, the owner may choose to reshuffle (offline) the whole database in order to physically remove the deleted pages. Page modifications are handled as regular queries, i.e., they can either produce a cache hit (if the page is stored inside the cache) or a cache miss. In any case, the original page is replaced with the new version.

To handle insertion operations, the secure hardware should reserve in advance sufficient storage space in its internal data structures. Therefore, during the initial reshuffling stage, the secure hardware should create numerous dummy pages that may be utilized to store the newly inserted pages. These pages are marked as *deleted*, so pages that are explicitly deleted by the data owner may serve the same purpose as well. When a new page is created in the database, the secure hardware accesses the next block of k pages as usual. However, the $(k+1)$-th page is always a *deleted* page. The newly inserted page is then stored inside the cache, replacing one of the pages therein. Finally, the deleted page swaps positions with one of the k locations of the retrieved block, and the evicted page is copied over the deleted page.

5 Secure Hardware Deployment

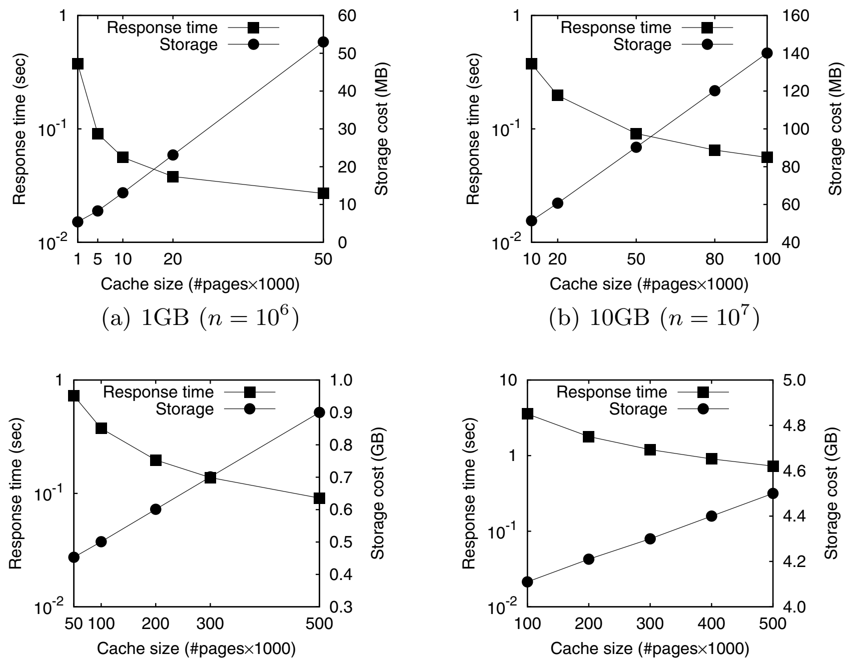
In this section we analyze the storage requirements and query processing cost of our methods in a secure hardware deployment. Our analysis is based on the configuration shown in Table 2, which is similar to the ones assumed in related studies [23,25].

Table 2. System specifications

Parameter	**Value**
Secure hardware cache	64MB
Disk seek time (t_s)	$5ms$
Disk read/write (r_d)	100 MB/s
Secure hardware link bandwidth (r_b)	80 MB/s
Encryption/decryption (r_{ed})	10 MB/s

Secure Storage Requirements. The page retrieval algorithm necessitates the storage of the three vectors depicted in Figure 2, inside the secure hardware cache. Given a database of n pages, each of size B bytes, the *pageCache* vector stores exactly m pages, thus consuming $m \cdot B$ bytes. The *serverBlock* vector stores the $k+1$ pages that are read from the server, i.e., it requires $(k+1) \cdot B$ bytes of space. Finally, the *pageMap* vector maintains information about the position of all database pages, plus an additional bit that indicates whether a page is currently cached. Consequently, it requires $n \cdot (\log n + 1)$ bits of storage space. Summarizing, the total storage cost of our approach (in bytes) is given as:

$$S = n \cdot \left\lceil \frac{(\log n + 1)}{8} \right\rceil + (m + k + 1) \cdot B \qquad (7)$$



(a) 1GB ($n = 10^6$)

(b) 10GB ($n = 10^7$)

(c) 100GB ($n = 10^8$)

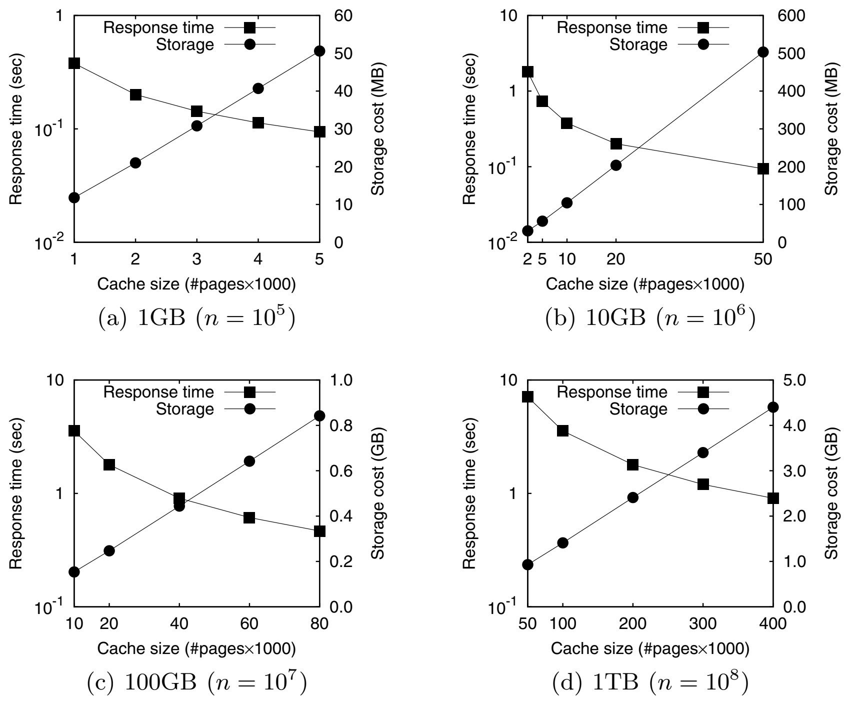
(d) 1TB ($n = 10^9$)

Fig. 4. Page retrieval costs for 1KB pages ($c = 2$)

Page Retrieval Cost. For every client query the secure hardware needs to perform 4 random accesses at the server's disk. Two of those correspond to the read operations (one for reading the next block, and one for the additional page), while the remaining two are performed for writing back the re-encrypted pages. The $k+1$ accessed pages are transferred twice between the secure hardware and the server (read/write) and are also processed twice by the encryption/decryption circuitry inside the secure hardware. Therefore, the query processing time at the server for retrieving a single page from the disk is:

$$Q_t = 4 \cdot t_s + 2 \cdot (k+1) \cdot B \cdot \left(\frac{1}{r_d} + \frac{1}{r_b} + \frac{1}{r_{ed}} \right) \tag{8}$$

Figures 4 and 5 show some sample configurations for retrieving 1KB and 10KB pages, respectively, from databases of different sizes (with a privacy parameter $c = 2$). Specifically, they depict the page retrieval times and storage space requirements at the secure hardware as a function of the cache size m. For a 1GB database, a single secure coprocessor can retrieve privately 1KB pages in $27ms$ and 10KB pages in $94ms$. Note that, unlike existing secure hardware PIR schemes that feature *amortized* cost, the processing times shown here are *constant*. For larger databases, we may leverage multiple coprocessors at the server site to increase the secure storage capacity. This will boost the value of m, thus



(a) 1GB ($n = 10^5$) (b) 10GB ($n = 10^6$)

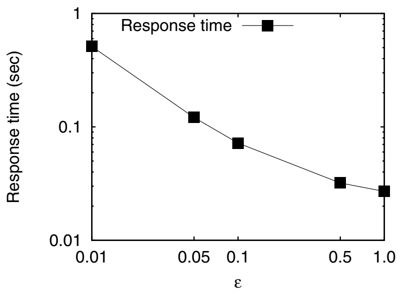
(c) 100GB ($n = 10^7$) (d) 1TB ($n = 10^8$)

Fig. 5. Page retrieval costs for 10KB pages ($c = 2$)

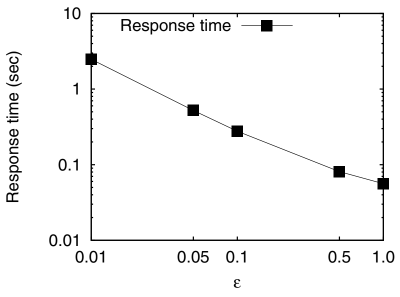
reducing considerably the security parameter k. For instance, with 1 coprocessor (up to 64MB of storage space) and a 10GB database, we can retrieve 1KB pages in 197ms and 10KB pages in 731ms. On the other hand, combining the storage space of 2 coprocessors can reduce those times to 65ms and 378ms, respectively.

Larger databases cannot be trivially handled by the current technology of tamper-resistant CPUs, due to the minimal storage resources that they provide. Consequently, 100GB databases will require 10 coprocessors to retrieve 1KB pages in 197ms and 10KB pages in 613ms. Even though this is an entirely feasible solution, it may increase considerably the monetary cost of PIR. Finally, for 1TB databases, sub-second page retrieval times (727ms for 1KB pages and 907ms for 10KB pages) are only feasible with over 4GB of secure storage. With the current technology, this capacity translates to over 70 coprocessor units. This excessive cost is mainly due to the *pageMap* data structure that maintains the location of every database page on the disk. However, this is an unavoidable cost because, unlike previous approaches that use hash functions to permute the entire database, our scheme reshuffles on a per page level and necessitates each page to be stored individually.

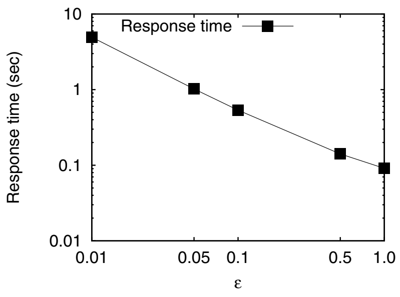
Figure 6 depicts the query response time as a function of the privacy parameter $c = 1 + \varepsilon$. We consider 1KB pages and set the cache sizes for the different databases to their largest values shown in Figure 4. Clearly, there is a



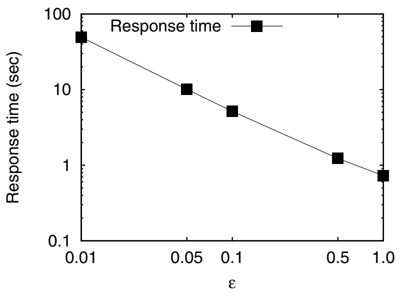
(a) 1GB ($m = 50000$)



(b) 10GB ($m = 100000$)



(c) 100GB ($m = 500000$)



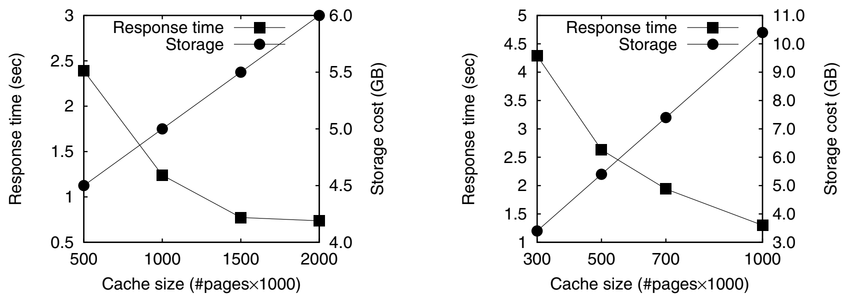
(d) 1TB ($m = 500000$)

Fig. 6. Response time as a function of $c = 1 + \varepsilon$ ($B = 1$KB)

trade-off between the privacy level of our approach and the computational cost. If we wish to provide better privacy, we need to retrieve more pages per block (increase k) in order to reduce the value of T. As shown in Equation (5), this will essentially decrease the value of the privacy parameter c. Nevertheless, our algorithm is efficient under strict privacy requirements and, for databases up to 100GB, sub-second query response times are achievable even for $c = 1.1$.

Despite the restrictions of current secure hardware technology, our methods are also applicable in the two-party querying model, as explained in Section 3.1. In this setting, the functionality of the secure hardware can be implemented on a powerful server (physically isolated from the adversary), thus allowing for much larger cache sizes. Consequently, our page retrieval algorithm can be implemented efficiently even for TB-sized databases. To verify the efficiency of this approach, we measured the page retrieval costs from a real implementation[2] of the two-party model. We set up the service provider and the owner to run on two different machines that were connected through a WiFi network. The network round-trip time (RTT) was set to $50ms$ and was simulated with the `sleep` function. Figure 7 illustrates the query response time and storage cost at the data

[2] We used the `Boost.Asio` library for the networking primitives and the `Crypto++` library for the AES implementation.



(a) 1KB pages ($n = 10^9$) (b) 10KB pages ($n = 10^8$)

Fig. 7. Page retrieval costs for 1TB database ($c = 2$)

owner as a function of the cache size m. With 6GB of storage space, the system can accommodate 2 million pages in its cache, achieving a query response time of $0.737ms$ (for 1KB pages). Note that the bottleneck in this architecture is the network transfer cost, since our algorithm necessitates the transfer of $(k + 1)$ database pages *twice* between the owner and the service provider. As a result, retrieving larger pages (10KB) requires a significant amount of storage space (to reduce the value of the security parameter k) and, as shown in Figure 7(b), over 10GB of space is necessary to achieve a query response time of $1.3s$.

6 Conclusions

Privacy preserving query processing is an emerging research field in the database community, due to the increasing demand for protecting user privacy. Existing techniques fail to provide adequate solutions, because they do not achieve a good trade-off between computational cost and privacy guarantees. On one hand, anonymity and encryption based schemes are computationally efficient, but they provide weak privacy. On the other hand, private information retrieval techniques offer perfect privacy, but their high computational cost renders them impractical for large databases. In this paper, we introduce a novel approach that provides a much stronger notion of privacy compared to anonymity or encryption based schemes, but with a computational cost that is considerably lower compared to existing PIR approaches. Our methods are built on top of a secure hardware that acts as a proxy between the clients and the server. The secure hardware encrypts and constantly reshuffles the database pages, in order to create sufficient uncertainty regarding the exact location of an arbitrary page. We give a formal definition of the privacy level of our algorithm and illustrate how to apply it in practice. Based on the performance characteristics of the current state-of-the-art secure hardware platforms, we show that our method is computationally efficient, even for very large databases.

Acknowledgments. This research has been funded by the NSF CAREER Award IIS-0845262.

References

1. Agrawal, D., Abbadi, A.E., Emekçi, F., Metwally, A.: Database management as a service: Challenges and opportunities. In: ICDE (2009)
2. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: SIGMOD (2004)
3. Ardagna, C.A., Cremonini, M., Damiani, E., di Vimercati, S.D.C., Samarati, P.: Location privacy protection through obfuscation-based techniques. In: DBSec (2007)
4. Barbaro, M., Zeller, T.: A face is exposed for AOL searcher no. 4417749. The New York Times (August 9, 2006)
5. Beimel, A., Ishai, Y., Kushilevitz, E., Raymond, J.E.: Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. In: FOCS (2002)
6. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)
7. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: FOCS (1995)
8. Duckham, M., Kulik, L.: Simulation of obfuscation and negotiation for location privacy. In: Cohn, A.G., Mark, D.M. (eds.) COSIT 2005. LNCS, vol. 3693, pp. 31–48. Springer, Heidelberg (2005)
9. Garrett, P.: Making, Breaking Codes: Introduction to Cryptology, 1st edn. Prentice-Hall, Englewood Cliffs (2001)
10. Gentry, C., Ramzan, Z.: Single-database private information retrieval with constant communication rate. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 803–815. Springer, Heidelberg (2005)
11. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., Tan, K.L.: Private queries in location based services: Anonymizers are not necessary. In: SIGMOD (2008)
12. Goldberg, I.: Improving the robustness of private information retrieval. In: IEEE Symposium on Security and Privacy (2007)
13. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. Journal of the ACM 43(3), 431–473 (1996)
14. Iliev, A., Smith, S.: Private information storage with logarithmic-space secure hardware. In: i-NetSec (2004)
15. Jones, R., Kumar, R., Pang, B., Tomkins, A.: I know what you did last summer: Query logs and user privacy. In: CIKM (2007)
16. Kalnis, P., Ghinita, G., Mouratidis, K., Papadias, D.: Preventing location-based identity inference in anonymous spatial queries. TKDE 19(12), 1719–1733 (2007)
17. Khoshgozaran, A., Shahabi, C., Shirani-Mehr, H.: Location privacy: Going beyond k-anonymity, cloaking and anonymizers. In: KAIS (2010)
18. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: FOCS (1997)
19. Lipmaa, H.: An oblivious transfer protocol with log-squared communication. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)
20. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The New Casper: Query processing for location services without compromising privacy. In: VLDB (2006)
21. Murugesan, M., Clifton, C.: Providing privacy through plausibly deniable search. In: SDM (2009)

22. Pang, H., Ding, X., Xiao, X.: Embellishing text search queries to protect user privacy. PVLDB 3(1), 598–607 (2010)
23. Papadopoulos, S., Bakiras, S., Papadias, D.: Nearest neighbor search with strong location privacy. PVLDB 3(1), 619–629 (2010)
24. Wang, S., Ding, X., Deng, R.H., Bao, F.: Private information retrieval using trusted hardware. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 49–64. Springer, Heidelberg (2006)
25. Williams, P., Sion, R.: Usable PIR. In: NDSS (2008)
26. Williams, P., Sion, R., Carbunar, B.: Building castles out of mud: Practical access pattern privacy and correctness on untrusted storage. In: CCS (2008)
27. Woodruff, D.P., Yekhanin, S.: A geometric approach to information-theoretic private information retrieval. In: IEEE Conference on Computational Complexity (2005)

Purpose Control: Did You Process the Data for the Intended Purpose?*

Milan Petković[1,3], Davide Prandi[2], and Nicola Zannone[3]

[1] Philips Research Eindhoven
milan.petkovic@philips.com
[2] Centre for Integrative Biology, University of Trento
prandi@science.unitn.it
[3] Eindhoven University of Technology
n.zannone@tue.nl

Abstract. Data protection legislation requires personal data to be collected and processed only for lawful and legitimate purposes. Unfortunately, existing protection mechanisms are not appropriate for purpose control: they only prevent unauthorized actions from occurring and do not guarantee that the data are actually used for the intended purpose. In this paper, we present a flexible framework for purpose control, which connects the intended purpose of data to the business model of an organization and detects privacy infringements by determining whether the data have been processed only for the intended purpose.

1 Introduction

In recent decades, many countries have enacted privacy laws and regulations that impose very stringent requirements on the collection and processing of personal data (e.g., EU Directive 95/46/EC, HIPAA). Purpose control plays a central role in such legislation [1]: personal data shall be collected for specified, lawful and legitimate purposes and not processed in ways that are incompatible with the purposes for which data have been collected. Purpose control requires the deployment of mechanisms that hold users accountable for their actions by verifying how data have actually been processed.

In contrast, current security and data protection mechanisms do not provide appropriate support for purpose control. They are preventive and, more importantly, they do not check for which purpose data are processed after access to data has been granted. Protection of personal information is often implemented by augmenting access control systems with the concept of purpose [2,3,4,5] (hereafter, we call access control policies augmented with purpose *data protection policies*). Here, protecting data implies guaranteeing that data are disclosed solely to authorized users with the additional condition that data are requested for the intended purpose. The access purpose is usually specified by the requester [4], implying complete trust on users. This poses risks of re-purposing the data [6,7] as users might process the data for purposes other than those for which the data were originally obtained. Therefore, to ensure compliance to data protection

* This work has been partially funded by the EU-IST-IP-216287 TAS^3 project.

© Springer-Verlag Berlin Heidelberg 2011

and purpose control, it is necessary to extend the current preventive approach by implementing mechanisms for verifying the actual use of data.

Some privacy enhancing technologies (e.g., [2,8]) partially address this issue. They collect and maintain *audit trails*, which record the actual user behavior, for external privacy audits. These auditing activities, however, are usually manual; the auditors sample and inspect the audit trails recorded by the system. The lack of systematic methods for determining how data are used makes auditing activities time-consuming and costly. For instance, at the Geneva University Hospitals, more than 20,000 records are opened every day [9]. In this setting, it would be infeasible to verify every data usage manually, leading to situations in which privacy breaches remain undetected.

In this paper, we present a framework for purpose control which detects privacy infringements by determining whether data are processed in ways that are incompatible with the intended purpose of data. To this end, we need a purpose representation model that connects the intended purpose of data to the business activities performed by an organization and methods able to determine whether the data are actually processed in accordance with purpose specifications.

Organizations often make use of business processes to define how organizational goals should be fulfilled. These *organizational processes* define the expected user behavior for achieving a certain organizational goal. The idea underlying our approach is to link the purposes specified in data protection policies to organizational goals and, therefore, to the business processes used to achieve such goals. Thus, the problem of verifying the compliance of data usage with the intended purpose consists in determining whether the audit trail is a valid execution of the organizational processes representing the purposes for which data are meant to be used. Intuitively, if the audit trail does not correspond to a valid execution of those processes, the actual data usage is not compliant with the purpose specification.

To enable automated analysis, we use the Calculus of Orchestration of Web Services (COWS) [10] for the representation of organizational processes. COWS is a foundational calculus strongly inspired by WS-BPEL [11]. It is based on a very small set of primitives associated with a formal operational semantics that can be exploited for the automated derivation of the dynamic evolution of the process. The COWS semantics makes it possible to construct a labeled transition system that generates the set of traces equivalent to the set produced by all possible executions of the process.

A naïve approach for purpose control would be to generate the transition system of the COWS process model and then verify if the audit trail corresponds to a valid trace of the transition system. Unfortunately, the number of possible traces can be infinite, for instance when the process has a loop, making this approach not feasible. Therefore, in this paper we propose an algorithm that replays the audit trail in the process model to detect deviations from the expected behavior. We demonstrate that the algorithm terminates and is sound and complete.

The structure of the paper is as follows. Next section presents our running example. (§2). Then, we introduce the building blocks of our framework and analyze their alignment (§3). We present an algorithm for purpose control (§4) and demonstrate the termination, soundness and completeness of the algorithm (§5). Finally, we discuss related work (§6) and conclude the paper, providing directions for future work (§7).

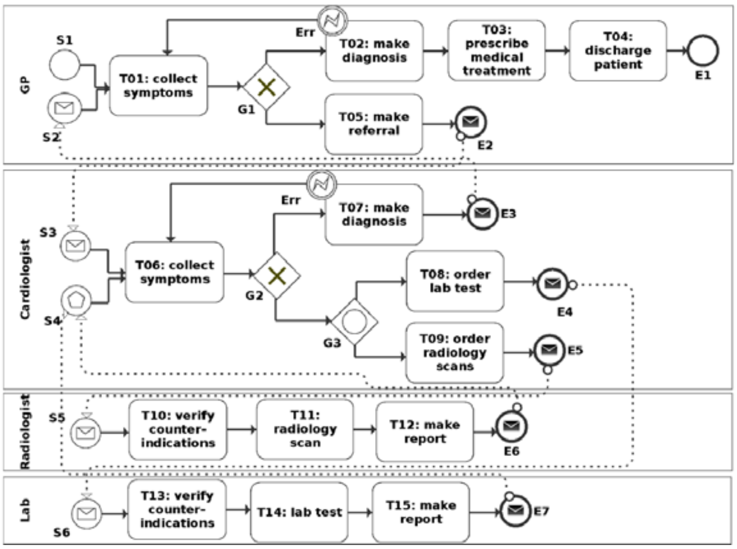


Fig. 1. Healthcare Treatment Process

2 Running Example

This section presents a simple scenario in the healthcare domain to illustrate our approach. Consider a patient who goes to a hospital to see a doctor. The hospital is equipped with its own Hospital Information System (HIS) to manage the administrative, financial, and clinical aspects of patient information. In particular, patient information are stored in an electronic patient record (EPR); here, we assume that EPRs are organized in sections; each of them contains a certain type of information. Members of the hospital staff can access specific sections of the EPR (or parts of them) depending on their position within the hospital and for well defined purposes. Suppose that a patient, Jane, did not give the hospital consent to process her information for research purposes. Then, the hospital staff cannot access Jane's information for clinical trials.

The provision of healthcare treatments can be seen as a process that involves several parties. Fig. 1 describes an example of a healthcare treatment process specified in the Business Process Modeling Notation (BPMN) [12]. Here, every BPMN pool[1] represents the visit of the patient to a member of the clinical staff at the local hospital. The process starts when a patient visits the general practitioner (GP) at the hospital (S1). The GP accesses the HIS to retrieve the patient's EPR and makes a physical examination to collect the symptoms (T01). Based on the gathered information, the GP may suspect that the patient is affected by a certain disease. He can either make a diagnosis (T02) or refer to a specialist if the case is more complex (T05). For instance, in case the GP suspects a cardio-vascular disease, he can refer the patient to a cardiologist.

[1] In BPMN a pool is used to specify the boundaries of the activities to be performed by a participant in the process and is graphically represented as a container enclosing those activities.

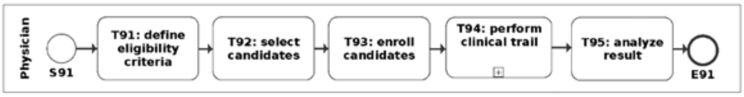


Fig. 2. Clinical Trial Process

If the patient is referred to a cardiologist (S3), the cardiologist accesses patient medical history in the HIS and makes a medical examination to collect the symptoms (T06). Based on this information, the cardiologist can either make a diagnosis directly (T07), or request lab tests or radiology scans (T08 and T09, respectively). If the resulting tests and scans are not good or a diagnosis cannot be made based on them, further tests can be required. When the lab or the radiology department receive the request for tests from the cardiologist (S5 and S6, respectively), they check the EPR for allergies or other counter-indications (T10 and T03, respectively). Then, they do the lab exam (T11 and T14) and export the results to the HIS (T12 and T15). A notification is sent to the cardiologist when the tests or the scans have been completed (E6 and E7).

When the cardiologist receives a notification for all the ordered tests and scans (S4), he retrieves the test results or the scans from the HIS (T06) and, based on them, makes a diagnosis (T07). After the cardiologist enters the diagnosis into the HIS, a notification is sent to the GP (E4). Upon the notification (S2), the GP extracts the diagnosis made by the cardiologist, scans and tests from the patient's EPR (T01). Based on them, the GP prescribes medical treatments to the patient (T03) and discharges her (T04).

Suppose now that the same cardiologist mentioned above is involved in a clinical trial. Fig. 2 shows the part of the clinical trial process in which a physician is involved. The physician has to define eligibility criteria for the trial (T91). Then, he accesses EPRs to find patients with a specific condition that meet the eligibility criteria (T92). In the next step, the physician asks the selected candidates whether they want to participate in the trial (T93). The trial is then performed (T94) and the results are analyzed (T95). Note that the cardiologist can get access to patient information for the legitimate purpose (i.e., claiming that it is for healthcare treatment) and then use the data for research purposes (i.e., clinical trial). Preventive mechanisms are not able to cope with these situations. In particular, they cannot prevent a user to process data for other purposes after the same user has legitimately got access to them.

To address this issue, we need mechanisms that make users accountable for their actions by determining how data were actually used. In this paper, we propose an approach that enables purpose control by determining if an audit trail is a valid execution of the process used by the organization to implement the intended purposes. In our scenario, this implies verifying that every usage of patient information is part of the sequence of tasks that the cardiologist and the other parties involved in the provision of healthcare treatments have to perform in order to accomplish the goal.

3 A Framework for Purpose Control

In this section, we propose a formal framework for purpose control. The main goal of the framework is to verify if a user processed the data only for the intended purpose. The proposed approach uses and interlinks the following three components:

- *Data protection policies* that define who can access the data and for which purpose.
- *Organizational processes* that describe the business processes and procedures of an organization.
- *Audit trails* that record the sequence of actions performed by users.

In the remainder of this section, we present the formal model components of the framework and discuss their alignment.

3.1 Basic Concepts and Notation

Our framework includes sets of five basic elements: users ($\mathcal{U}$), roles ($\mathcal{R}$), objects ($\mathcal{O}$), actions ($\mathcal{A}$), and organizational processes ($\mathcal{P}$). A *user* denotes any entity capable to perform actions on an object, and a *role* denotes a job function describing the authority and responsibility conferred on a user assigned to that role. We assume that roles are organized into a hierarchical structure under partial ordering $\geq_R$. Such an order reflects the principles of generalization and specialization. Let $r_1, r_2 \in \mathcal{R}$ be roles. We say that r_1 is a specialization of r_2 (or r_2 is a generalization of r_1) if $r_1 \geq_R r_2$.

Objects denote resources (i.e., data, services, or system components) under protection. We use a directory-like notation to represent hierarchical resources like file systems and EPRs. This implies a partial ordering $\geq_O$ on resources, which reflects the data structure. In addition, we make explicit the name of the data subject when appropriate. For instance, Jane's EPR is denoted as $[Jane]EPR$, whereas $[Jane]EPR/Clinical$ denotes the section containing medical information in her EPR, with $[Jane]EPR \geq_O [Jane]EPR/Clinical$. We use $[\cdot]EPR$ to denote EPRs, regardless a specific patient.

An *action* is an operation that can be performed on a resource. The set of actions $\mathcal{A}$ includes "read", "write", and "execute".

Organizational processes specifies the sequences of *tasks* that have to be performed to achieve organizational goals. We use $\mathcal{Q}$ to denote the set of tasks belonging to any process in $\mathcal{P}$. Process models can specify the entity that is expected to perform a certain task (e.g., using pools in BPMN). However, processes usually are not specified in terms of identifiable users. For example, a hospital would not define a specific clinical trial process for each physician. Business processes are intended to be general in order to cover a large number of scenarios. Therefore, we assume that every BPMN pool corresponds to a role in $\mathcal{R}$. Finally, a given task can be part of more than one process, and several instances of the same process can be executed concurrently. To apply conformance checking techniques, it is necessary to distinguish the process instance (the so called *case*) in which tasks are performed [13]. Hereafter, $\mathcal{C}$ denotes the set of cases.

In data protection, the concept of *purpose* plays a central role [1,2,4]. The purpose denotes the reason for which data can be collected and processed. In particular, we refer to purposes that regulate data access as *intended purposes*, and to purposes for which data access is requested as *access purposed*. Purposes are related to the business activities of an organization and can be identified with organizational goals. For instance, [14] defines a list of purposes for healthcare, which includes healthcare treatment, payment, research and marketing. Thus, in this paper, we represent purposes by the organizational process implemented by an organization to achieve the corresponding organizational goal.

$(Physician, read, [\cdot]EPR/Clinical, treatment)$
$(Physician, write, [\cdot]EPR/Clinical, treatment)$
$(Physician, read, [\cdot]EPR/Demographics, treatment)$
$(MedicalTech, read, [\cdot]EPR/Clinical, treatment)$
$(MedicalTech, read, [\cdot]EPR/Demographics, treatment)$
$(MedicalLabTech, write, [\cdot]EPR/Clinical/Tests, treatment)$
$(Physician, read, [X]EPR, clinicaltrial)$

Fig. 3. Sample Data Protection Policy

3.2 Data Protection Policies

The aim of data protection policies is to protect an individual's right to privacy by keeping personal data secure and by regulating its processing. Several languages for the specification of data protection policies have been proposed in literature [3,4]. The objective of this paper is not to propose yet another policy specification language. Thus, here we present a simple language that suffices for the purpose of this paper.

A data protection policy specifies the access rights: who is authorized to access the system, what actions he is allowed to perform on a resource, and for which purpose.

Definition 1. *A* data protection statement *is a tuple* (s, a, o, p) *where* $s \in \mathcal{U} \cup \mathcal{R}$*,* $a \in \mathcal{A}$*,* $o \in \mathcal{O}$*, and* $p \in \mathcal{P}$*. A* data protection policy Pol *is a set of data protection statements.*

Fig. 3 presents the data protection policy governing our scenario. The first block of Fig. 3 states that physicians can read and write patient medical information in EPRs for treatment. Moreover, physicians can also read patient demographics for treatment. Note that roles GP, radiologist, and cardiologist are specializations of role physician. The second block of statements targets medical technicians. They can read patient medical information for treatment. Moreover, medical lab technicians (which is a specialization of medical technicians) have write permission on the EPR section concerning test results. The last block of Fig. 3 represents the hospital policy that allows physicians to access the EPR of those patients (X) who give consent to use their data for clinical trial.

Users can request access to system's resources by means of access request.

Definition 2. *An* access request *is a tuple* (u, a, o, q, c) *where* $u \in \mathcal{U}$*,* $a \in \mathcal{A}$*,* $o \in \mathcal{O}$*,* $q \in \mathcal{Q}$*, and* $c \in \mathcal{C}$*.*

An access request specifies the user who makes the request, the object to be accessed, and the action to be performed on the object along with information about the purpose for which the request is made. In particular, the access purpose is represented by the task for which the access to the object is requested and by the process instance.

When a user requests permission to execute an action on an object for a certain purpose, the access request is evaluated against the data protection policy. Access is granted if there exists a data protection statement that matches the access request directly or through a hierarchy.

Definition 3. *Let* Pol *be a data protection policy and* (u, a, o, q, c) *an access request. We say that the* access request is authorized *if there exists a statement* $(s, a', o', p) \in Pol$ *such that (i)* $s = u$*, or* $s = r_1$*,* u *has role* r_2 *active,*[2] *and* $r_2 \geq_R r_1$*; (ii)* $a = a'$*; (iii)* $o' \geq_O o$*; (iv)* c *is an instance of* p*, and* q *is a task in* p*.*

[2] We assume users have to authenticate within the system before performing any action. During the authentication process, the role membership of users is determined by the system.

3.3 Organizational Processes

Organizational processes specify the activities that users are expected to take in order to accomplish a certain goal. Organizational processes are often modeled in BPMN [12], the de-facto standard modeling notation in industry. Although BPMN provides a standard visualization mechanism, it is informal and therefore not suitable for formal verification. We rely on COWS [10], a foundational language for service-oriented computing that combines constructs from process calculi with constructs inspired by WS-BPEL [11], for the formalization of BPMN processes. Here, we present a minimal version of COWS that suffices for representing organizational processes.

COWS basic entities are *services*, i.e., structured activities built by combining basic activities. COWS relies on three countable and pairwise disjoint sets: *names*, *variables*, and *killer labels*. Basic activities take place at *endpoints*, identified by both a *partner* and an *operation* name. The grammar of COWS is defined as follows:

$$\begin{aligned} s &::= p \cdot o!\,\langle w \rangle \;\big|\; [\,d\,]s \;\big|\; g \;\big|\; s \mid s \;\big|\; \{\!|s|\!\} \;\big|\; \mathbf{kill}(k) \;\big|\; *\, s \\ g &::= \mathbf{0} \;\big|\; p \cdot o?\,\langle w \rangle .\, s \;\big|\; g + g \end{aligned}$$

Intuitively, the basic activities a service can perform are: the empty activity $\mathbf{0}$; $p \cdot o!\,\langle w \rangle$, an *invoke* (sending) activity over endpoint $p \cdot o$ with parameter w; $p \cdot o?\,\langle w \rangle$, a *request* (receiving) activity over endpoint $p \cdot o$ with parameter w; $\mathbf{kill}(k)$, a *block* activity that prevents services within the scope of a killer label k to proceed. The scope for names, variables, and killer labels is denoted by $[\,d\,]s$. The construct $\{\!|s|\!\}$, when not covered by an action, saves a service s from a killer signal sent out by a $\mathbf{kill}(\_)$.

The temporal execution order of the basic activities is described by a restricted set of operators: $p \cdot o?\,\langle w \rangle .\, s$ executes request $p \cdot o?\,\langle w \rangle$ and *then* service s; services s_1 and s_2 running in *parallel* are represented as $s_1 | s_2$; a *choice* between two request activities g_1 and g_2 is written as $g_1 + g_2$. Finally, recursion is modeled with the replication operator $*$: the service $*\, s$ behaves as $*\, s \mid s$, namely $*\, s$ spawns as many copies of s as needed.

COWS is equipped with a structural operational semantics [15], i.e., a set of syntax-driven axioms and rules which describes the dynamics of the system at hand. Specifically, rules allow the definition of a *Labeled Transition System* (LTS) $(s_0, S, L, \rightarrow)$, where S is a set of COWS services or *states*, $s_0 \in S$ is the *initial state*, L is a set of *labels*, and $\rightarrow \subseteq S \times L \times S$ is a *labeled transition relation* among COWS services, such that $(s, l, s') \in \rightarrow$ iff COWS semantics allows one to infer the labeled transition. We use $s \xrightarrow{l} s'$ as a shortcut for $(s, l, s') \in \rightarrow$.

Labels in L are generated by the following grammar:

$$l ::= (p \cdot o) \lhd w \;\big|\; (p \cdot o) \rhd w \;\big|\; p \cdot o\,(v) \;\big|\; \dagger k \;\big|\; \dagger$$

Invoke label $(p \cdot o) \lhd w$ and request label $(p \cdot o) \rhd w$ are for invoke and request activities, respectively. Label $p \cdot o\,(v)$ represents a communication between an invoke label $(p \cdot o) \lhd v$ and a request label $(p \cdot o) \rhd w$. If the communication is indeed a synchronization, the label is simplified as $p \cdot o$. Finally, labels $\dagger k$ and $\dagger$ manage, respectively, ongoing and already executed killing activities. We refer to [10] for a detailed description of the COWS structural operational semantics.

The encoding of BPMN processes into COWS specifications founds on the idea of representing every BPMN element as a distinct COWS service. For the lack of space, here we only present the intuition of the encoding; examples of the encoding are given in Appendix A. In [16], we have defined elementary and parametric COWS services for a core set of BPMN elements. For example, a start event (e.g., $S1$ in Fig. 1) is modeled in COWS as $x \cdot y!\,\langle\rangle$ where $x \cdot y$ is the endpoint triggering the next BPMN element in the flow (x is the pool that contains the element and y is the name of the element); a task (e.g., $T01$ in Fig. 1) is modeled as $x \cdot y?\,\langle\rangle.\,Act$, where $x \cdot y$ is the endpoint to trigger the execution of the task, and Act is the activity performed by the task (Act eventually specifies the next BPMN element). Parameters are instantiated to connect the BPMN elements forming the BPMN process. The COWS service implementing $S1$ and $T01$, denoted by $[[S1]]$ and $[[T01]]$ respectively, are $[[S1]] = GP \cdot T01!\,\langle\rangle$ and $[[T01]] = GP \cdot T01?\,\langle\rangle.\,[[Act]]$, where $[[Act]]$ is the COWS service implementing activity Act and GP stands for general practitioner. The overall organizational process results from the parallel composition of the elementary services implementing the single BPMN elements. The process composed by services $[[S1]]$ and $[[T01]]$ is $[[S1]] \mid [[T01]]$.

The sequence flow, which describes the execution order of process activities by tracking the path(s) of a token through the process, is rendered as a sequence of communications between two services. For instance, the sequence flow between event $S1$ and task $T01$ is defined by the labeled transition

$$[[S1]] \mid [[T01]] \xrightarrow{GP \cdot T01} \mathbf{0} \mid [[Act]]$$

Intuitively, the label $GP \cdot T01$ allows one to “observe” on the COWS transition system that the task received the token. The same idea applies to message flow as well as to other BPMN elements like event handlers and gateways. However, in case of event handlers and gateways, some “internal computation” is required to determine the next BPMN element to be triggered. For instance, exclusive gateway $G1$ in Fig. 1 is connected to $T02$ and $T05$; from $G1$, the token can flow either to $T02$ or $T05$, but not to both. The act of deciding which task should be triggered does not represent a flow of the token. In this case we use the private name sys as the partner in the label. Similarly, label $sys \cdot Err$ is used to represent error signals. In general, it is not known in advance how many times a service is invoked during the execution of a process. An example of this is given by cycles (e.g., $T01$, $G1$ and $T02$ in Fig. 1). To this end, we prefix COWS services with the replication operator $*$. This operator makes multiple copies of the COWS service; each copy corresponds to an invocation of the service.

3.4 Audit Trails

Auditing involves observing the actions performed by users to ensure that policies and procedures are working as intended or to identify violations that might have occurred. Audit trails are used to capture the history of system activities by representing events referring to the actions performed by users. Every event is recorded in a log entry.

Definition 4. *A* log entry *is a tuple* (u, r, a, o, q, c, t, s) *where* $u \in \mathcal{U}$, $r \in \mathcal{R}$, $a \in \mathcal{A}$, $o \in \mathcal{O}$, $q \in \mathcal{Q}$, $c \in \mathcal{C}$, t *ties an event to a specific time, and* s *is the task status indicator.*

The field *user* represents the user who performed the *action* on the *object*. *Role* represents the role held by the user at the *time* the action was performed. The *task status indicator* specifies whether the task succeeded or not (i.e., $s \in \{success, failure\}$). We assume that the failure of a task makes the task completed; therefore, no actions within the task are possible after the task has failed. In addition, the process can proceed only if there is in place a mechanism to handle the failure. Log entries also contain information about the purpose for which the action was performed. In particular, the purpose is described by the *task* in which the action was performed and the *case* that identifies the process instance in which the action took place.

An audit trail consists of the chronological sequence of events that happen within the system.

Definition 5. *A* audit trail *is an ordered sequence of log entries where given two entries* $e_i = (u_i, r_i, a_i, o_i, q_i, c_i, t_i, s_i)$ *and* $e_j = (u_j, r_j, a_j, o_j, q_j, c_j, t_j, s_j)$ *we say that* e_i *is before* e_j *(denoted by* $e_i < e_j$*) if* $t_i < t_j$*.*

Recent data protection regulations in the US (see [17]) impose healthcare providers to record all actions related to health information. Accordingly, we assume that audit trails record every action performed by users, and these logs are collected from all applications in a single database with the structure given in Def. 4. In addition, audit trails need to be protected from breaches of their integrity. A discussion on secure logging is orthogonal to the scope of this paper. Here, we just mention that there exist well-established techniques [18,19], which guarantee the integrity of logs.

Fig. 4 presents a possible audit trail for the scenario of Section 2. The audit trail describes the accesses to Jane's EPR made by the GP (John), the cardiologist (Bob), and the radiologist (Charlie) in the process of providing her medical treatments (we assume that Bob did not order lab tests). It also shows that a number of instances of the process can be executed concurrently. Time is in the form year-month-day-hour-minute. Tasks are denoted by a code as defined in Fig. 1. Case HT-1 represents the instance of the process being executed. In particular, HT stands for the healthcare treatment process and the number indicates the instance of the process.

The last part of Fig. 4 presents the log entries generated during the execution of the clinical trial (CT) process. Here, Bob specified healthcare treatment as the purpose in order to retrieve a larger number of EPRs.[3] Note that preventive mechanisms cannot detect the infringement. Only the patients selected for the trial might suspect a privacy breach. Certainly, the infringement remains covered for those patients who were not selected for the trial and did not allow the usage of their data for research purposes.

3.5 Alignment

In the previous sections, we introduced data protection policies, organizational processes, and audit trails. Although these components make use of the same concepts, such concepts are often specified at a different level of abstraction. In this section, we discuss how they are related to each other.

[3] Note that, if the physician specifies clinical trial as purpose, the HIS would only return the EPRs of those patients who gave their consent to use their information for research purposes.

user	role	action	object	task	case	time	status
John	GP	read	[Jane]EPR/Clinical	T01	HT-1	201003121210	success
John	GP	write	[Jane]EPR/Clinical	T02	HT-1	201003121212	success
John	GP	cancel	N/A	T02	HT-1	201003121216	failure
John	GP	read	[Jane]EPR/Clinical	T01	HT-1	201003121218	success
John	GP	write	[Jane]EPR/Clinical	T05	HT-1	201003121220	success
John	GP	read	[David]EPR/Demographics	T01	HT-2	201003121230	success
			...				
Bob	Cardiologist	read	[Jane]EPR/Clinical	T06	HT-1	201003141010	success
Bob	Cardiologist	write	[Jane]EPR/Clinical	T09	HT-1	201003141025	success
Charlie	Radiologist	read	[Jane]EPR/Clinical	T10	HT-1	201003201640	success
Charlie	Radiologist	execute	ScanSoftware	T11	HT-1	201003201645	success
Charlie	Radiologist	write	[Jane]EPR/Clinical/Scan	T12	HT-1	201003201730	success
Bob	Cardiologist	read	[Jane]EPR/Clinical	T06	HT-1	201003301010	success
Bob	Cardiologist	write	[Jane]EPR/Clinical	T07	HT-1	201003301020	success
John	GP	read	[Jane]EPR/Clinical	T01	HT-1	201004151210	success
John	GP	write	[Jane]EPR/Clinical	T02	HT-1	201004151210	success
John	GP	write	[Jane]EPR/Clinical	T03	HT-1	201004151215	success
John	GP	write	[Jane]EPR/Clinical	T04	HT-1	201004151220	success
Bob	Cardiologist	write	ClinicalTrial/Criteria	T91	CT-1	201004151450	success
Bob	Cardiologist	read	[Alice]EPR/Clinical	T06	HT-10	201004151500	success
Bob	Cardiologist	read	[Jane]EPR/Clinical	T06	HT-11	201004151501	success
			...				
Bob	Cardiologist	read	[David]EPR/Clinical	T06	HT-20	201004151515	success
Bob	Cardiologist	write	ClinicalTrial/ListOfSelCand	T92	CT-1	201004151520	success
Bob	Cardiologist	read	[Alice]EPR/Demographics	T06	HT-21	201004151530	success
			...				
Bob	Cardiologist	read	[David]EPR/Demographics	T06	HT-30	201004151550	success
Bob	Cardiologist	write	ClinicalTrial/ListOfEnrCand	T93	CT-1	201004201200	success
Bob	Cardiologist	write	ClinicalTrial/Measurements	T94	CT-1	201004221600	success
			...				
Bob	Cardiologist	write	ClinicalTrial/Measurements	T94	CT-1	201004291600	success
Bob	Cardiologist	write	ClinicalTrial/Results	T95	CT-1	201004301200	success

Fig. 4. Audit Trail

Audit trails usually capture and store information at a lower level of abstraction than organizational processes. While the basic components of organizational processes are tasks, the basic components of audit trails are actions. Accomplishing a task may require a user to execute a number of actions. Thereby, there is a 1-to-n mapping between tasks and log entries: one task can be associated with multiple log entries. One can think to bring organizational processes and audit trails at a comparable level of abstraction, for instance, by specifying a process in terms of actions or by annotating each task in the process with the actions that can be executed within the task. However, these approaches would require several efforts in the specification of organizational processes as well as affect their readability and understanding. To address this issue, we allow for every action executed within the tasks active at a certain time (when checking the compliance of the actual data usage with the intended purpose as described in Section 4). However, this leaves risks of unauthorized access to data. To prevent unauthorized access while keeping the management of organizational processes simple, a mechanism for purpose control should be complemented with a preventive enforcement mechanism that verifies access requests in isolation (i.e., independently from other requests).

Languages for representing business processes often rely on information that is not available in audit trails. For instance, the COWS representation of organizational processes founds on the idea that the evolution of the transition system completely characterizes the evolution of the process. Accordingly, transition systems contain information about the management of gateways and the occurrence of non-observable events, which is not recorded in audit trails. To define a suitable mapping between log entries and COWS labels, we distinguish the information that is IT observable by defining the set of *observable labels* $\overline{L}$ as a subset of the set of labels L, i.e., $\overline{L} \subset L$. In particular, labels in $\overline{L}$ specify labels representing the execution of a task q by a partner r (i.e., synchronization labels of the form $r \cdot q$) and labels representing errors (i.e., $sys \cdot Err$). Summing up, the set of observable labels is

$$\overline{L} = \{r \cdot q \mid r \in \mathcal{R} \text{ and } q \in \mathcal{Q}\} \cup \{sys \cdot Err\}.$$

Our definition of observable labels reflects the fact that we assume that only the execution of tasks and error events are IT observable. In case other activities can be logged by the system (e.g., message flows), the definition above should be extended to include the corresponding labels.

How the system determines the purpose for which an action was performed (i.e., the task and case in a log entry) is a critical issue as it is necessary to link the actions to the appropriate process instance. We assume that information about the purpose is available. Different solutions can be adopted to collect this information. For instance, most IT systems based on transactional systems such as WFM, ERP, CRM and B2B systems are able to record the task and the instance of the process [13]. Here, the system itself is responsible to determine the context of an action and store it in the log entry. A different approach is proposed in [2,3,4] where users are required to specify the purpose along with the access request. This approach is also adopted in existing EPR systems like DocuLive which require users to provide the reason for the access and record that reason in audit trails [20]. We assume that the purpose specified in the access request (i.e., q and c in Def. 2), which can be determined either by the user or by the system, is recorded in the corresponding log entry. In the next section, we present an algorithm for verifying if data have actually been processed for that purpose.

4 Compliance with Purpose Specification

The aim of purpose control is to guarantee that personal data are not processed in ways that are incompatible with the intended purpose of data. We identify two main issues to be addressed in order to ensure compliance to purpose control: (a) data access should be authorized, and (b) access purposes should be specified correctly and legally. The first issue is addressed by Def. 3 which provides a way to evaluate access request against data protection policies. In particular, this definition guarantees that the data requester has the permission necessary to access the data.

However, the real challenge in ensuring purpose control is to validate the access purpose specified in the access request, i.e. determining whether data were in fact processed for that purpose. In our scenario, the cardiologist legitimately accessed patient data for healthcare treatment and then used those data for clinical trial, which leads to

Algorithm 1. Compliance procedure

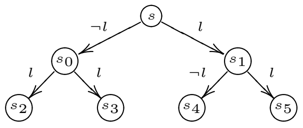
```
input  : a state $s$, an audit trail $l$
output: bool
1  let $conf\_set = \{(s, empty, \mathsf{WeakNext}(s))\}$;
2  let $next\_conf\_set = null$;
3  while $l \neq null$ do
4     let $l = e * l'$;
5     let $r \in \mathsf{R}$ s.t. $e.role \leq_R r$;
6     let $found = false$;
7     forall $conf \in conf\_set$ do
8        if $((r, e.task) \notin conf.active\_tasks) \vee (e.status = failure))$ then
9           forall $(label, state, active\_task) \in conf.next$ do
10             if $((label = r \cdot e.task) \wedge (e.status = success)) \vee ((label = sys \cdot Err) \wedge$
               $(e.status = failure))$ then
11                $found = true$;
12                $next\_conf\_set+ = (state, active\_task, \mathsf{WeakNext}(state))$;
13          end
14          else
15             $found = true$;
16             $next\_conf\_set+ = conf$;
17       let $l = l'$;
18       $conf\_set = next\_conf\_set$;
19       $next\_conf\_set = null$;
20    end
21    if $\neg found$ then return false;
22 end
23 return true;
```

a privacy infringement. To detect re-purposing of data, it is necessary to analyze the actual usage of data. Therefore, for each case in which the object under investigation was accessed, we determine if the portion of the audit trail related to that case is a valid execution of the process implemented by an organization to achieve the corresponding purpose using Algorithm 1.

The algorithm takes as input the COWS service representing the purpose and a finite (portion of) audit trail and determines whether the LTS associated to that service accepts the audit trail, i.e. the audit trail corresponds to a trace of the LTS. The key point of the algorithm is to determine if a log entry can be simulated at a certain point of the execution of the process. Hence we introduce the concept of configuration to represent the current state, the tasks active in that state, and the states reachable from the current state together with the set of active tasks in those states.

Definition 6. *Let S be the set of COWS services, $\mathcal{R}$ the set of roles, $\mathcal{Q}$ the set of tasks, and $\overline{L}$ the set of observable labels. A* configuration *$conf$ is a triple $(state, active\_tasks, next)$ where $state \in S$ represents the current state, $active\_tasks \in 2^{(\mathcal{R}\times\mathcal{Q})}$ represents the set of active tasks in the current state, and $next \in 2^{(\overline{L}\times S\times 2^{(\mathcal{R}\times\mathcal{Q})})}$ represents the possible states that can be reached from the current state executing $l \in \overline{L}$ together with the corresponding active tasks. Hereafter, we denote the components of a configuration as $conf.state$, $conf.active\_tasks$, and $conf.next$, respectively.*

The initial configuration consists of the state s representing a given process. Because a BPMN process is always triggered by a start event [12], the set of active tasks in the initial configuration is empty. The $conf.next$ is computed using function $\mathsf{WeakNext}$. This function takes state s as input and returns the set of states S reachable from s with exactly one observable label. Intuitively, this function explores the process and determines which activities can be executed and the states that are reachable executing such activities. Consider, for instance, the LTS in Fig. 5 where we use l for labels in $\overline{L}$ and $\neg l$ for labels in $L \setminus \overline{L}$. Function $\mathsf{WeakNext}(s)$ returns states s_1, s_2, and s_3. The reachable states are computed on the basis of the sequence and message flow. This requires analyzing the gateways which the token(s) goes through. For instance, parallel gateways (i.e., AND gateways) create parallel flow. Accordingly, $\mathsf{WeakNext}$ returns the states reachable following all the flows coming out from the gateway. On the other hand, inclusive decision gateways (i.e., OR gateways) are locations where the sequence flow can take one or more alternative paths. Therefore, the set of reachable states includes states that allow the execution of every possible combination of alternatives. For each reachable state, the function also computes the set of tasks active in that state. $\mathsf{WeakNext}$ can be implemented on top of CMC [21], an on-the-fly model checker and interpreter for COWS. This tool supports the derivation of all computations originating from a COWS process in automated way.



$$\begin{tikzcd} & & & s \arrow[dll, "\neg l"'] \arrow[drr, "l"] & & & \\ & s_0 \arrow[dl, "l"'] \arrow[dr, "l"] & & & & s_1 \arrow[dl, "\neg l"'] \arrow[dr, "l"] & \\ s_2 & & s_3 & & s_4 & & s_5 \end{tikzcd}$$

Fig. 5. WeakNext

Algorithm 1 extracts an entry e from the audit trail (line 4), and, for each configuration $conf$ in $next\_conf\_set$, it verifies whether the executed activity (i.e., $(r, e.task)$ where role r is a generalization of $e.role$ according to the role hierarchy) is active and succeeded. If it is the case, the task remains active and the configuration is added to the set of configurations to be considered in the next iteration ($next\_conf\_set$) (line 16). Otherwise, if the action succeeded and the execution of the task makes it possible to reach a reachable state, a new configuration for that state is created and added to $next\_conf\_set$. Similarly, if the activity failed and the failure leads to a reachable state, a new configuration for that state is created and added to $next\_conf\_set$.

The computation terminates with *false* (i.e., an infringement is detected), if the entry cannot be simulated by the process (line 21). Otherwise, the computation proceeds until the audit trail has been completely analyzed. If infringements are not detected, the computation terminates with *true* (line 23). Note that the analysis of the audit trail may lead the computation to a state for which further activities are still possible. In this case the analysis should be resumed when new actions within the process instance are recorded. However, if a maximum duration for the process is defined, an infringement can be raised in the case where this temporal constraint is violated.

We show now the application of Algorithm 1 to the process in Fig. 1 and the sequence of audit entries of Fig. 4 with case HT-1. Fig. 6 presents a portion of the transition system generated by the algorithm. Here, nodes represent the states visited by the algorithm together with the set of active tasks; edges represent observable transitions from the states. The number on the edge indicates the path of the transition system (i.e., the order of log entries).

It is worth noting that the failure of task $T02$ (step 3) leads to a state in which no tasks are active ($St4$). This state corresponds to a "suspension" of the process awaiting

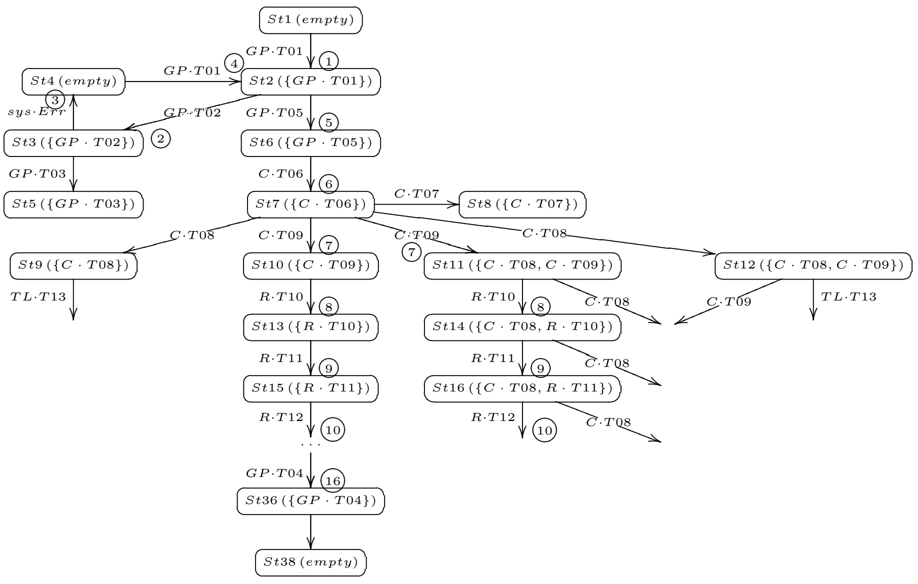


Fig. 6. Portion of the transition system visited by Algorithm 1

the proper activities ($GP \cdot T01$) to restore it. Moreover, one can notice that five states are reachable from state $St7$. This situation is due to the combination of exclusive decision gateway $G2$ and inclusive decision gateway $G3$ (see Fig. 1). In particular, $G3$ allows four possible states, one in which the cardiologist (C in Fig. 6) ordered only lab tests ($St9$), one in which he ordered only radiology scans ($St10$), and two in which he ordered both ($St11$ and $St12$). The difference between these two states lies in the activity executed to reach them. The next log entry refers to the radiologist verifying counter-indications ($R \cdot T10$). States $St10$ and $St11$ allow the execution of that activity; therefore, both states are considered in the next iteration. The algorithm proceeds until the audit trail is fully simulated by the process (step 17). As the portion of the audit trail corresponding to HT-1 is completely analyzed without deviations from the expected behavior, no infringement is detected by the algorithm.

Besides for HT-1, Jane's EPR has been accessed for case HT-11 (see Fig. 4). If we apply the algorithm to the portion of the audit log corresponding to that case (only one entry), we can immediately see that it does not correspond to a valid execution of the HT process. Therefore, the algorithm returns *false* indicating the presence of an infringement: Jane's EPR was accessed for healthcare treatment (HT-11), but it was not processed for that purpose. The cardiologist might have launched a mimicry attack to cover the infringement. However, achieving a purpose requires the execution of a number of tasks; each task should be executed by a user who is allowed to hold the role to which the task is assigned. Therefore, a single user cannot simulate the whole process alone, but he has to collude with other users to make a mimicry attack successful. Moreover, if the cardiologist reuses a previous case as the reason for the request (i.e., HT1 instead of HT11), the attack would succeed only in very restricted time windows:

the unlawful access has to be in conjunction with a legitimate access, otherwise Algorithm 1 recognizes that the audit trace is not a valid execution of the process. This threat can be partially mitigated by limiting multi-tasking, i.e. a user have to complete an activity before starting a new activity.

5 Properties of the Algorithm

In the previous section, we have proposed an algorithm that can either confirm that the data were processed for the intended purpose or detect privacy infringements. In this section, we discuss the termination, soundness and completeness of the algorithm. Proofs are presented in Appendix B.

The critical part of Algorithm 1 is function WeakNext, because, given a COWS service s, it has to generate and explore (part of) the possibly infinite transition system $LTS(s)$ associated to s; thereby, we have to guarantee the termination of WeakNext. We start by giving some basic definitions.

Given a LTS $\Omega = (s, S, L, \rightarrow)$, a trace is a possibly infinite sequence of $S \times L$ pairs $\sigma \equiv (s, l_0), (s_0, l_1) \ldots (s_n, l_{n+1}) \ldots$ describing a trajectory of the LTS, also denoted as $s \xrightarrow{l_0} s_0 \xrightarrow{l_1} \ldots s_n \xrightarrow{l_{n+1}} \ldots$. The possibly infinite set of traces of Ω is $\Sigma(\Omega)$.

A formal definition of the set of states computed by function WeakNext follows.

Definition 7. *Let s be a COWS service and $\overline{L}$ the set of observable labels for s. Then,* $\mathsf{WeakNext}(s) = \{s' \mid \exists k < \infty . s \xrightarrow{l_0} \ldots \xrightarrow{l_k} s_k \xrightarrow{l} s' \wedge \forall i \leq k \,.\, l_i \notin \overline{L} \wedge l \in \overline{L}\}$.

Given a COWS service s, function $\mathsf{WeakNext}(s)$ is decidable w.r.t. the set of observable labels $\overline{L}$ if, for each trace from s, it is not possible to perform an infinite number of transitions with label in $L \setminus \overline{L}$. The following generalizes this concept.

Definition 8. *Let $\Omega = (s, S, L, \rightarrow)$ be a LTS and $M \subseteq L$ a set of labels. A trace $\sigma \equiv (s, l_0), (s_0, l_1) \ldots (s_n, l_{n+1}) \ldots$ in $\Sigma(\Omega)$ is* finitely observable *w.r.t M iff $\exists n < \infty \,.\, l_n \in M$ and $\forall j > n \,.\, (l_j \in M \Rightarrow \exists k < \infty \,.\, l_{j+k} \in M)$. The set of finitely observable traces of Ω is denoted as $\Sigma^{FO}(\Omega)$. If $\Sigma(\Omega) = \Sigma^{FO}(\Omega)$, Ω is a* finitely observable LTS *w.r.t M.*

A finitely observable transition system w.r.t. a set of labels M could express infinite behaviors, but, within a finite time period it is possible to observe a label in M. This is the idea underlying Algorithm 1. In particular, given a task active at a certain step of the execution of the process, the algorithm determines what are the possible sets of active tasks in the next step. The definition of finitely observable transition system allows us to state a first important result.

Proposition 1. *Given a COWS service s and the set of observable labels $\overline{L}$ for s, if $LTS(s)$ is finitely observable w.r.t. $\overline{L}$, then* $\mathsf{WeakNext}(s)$ *is decidable on $\overline{L}$.*

Although COWS is expressive enough to represent BPMN processes, in order to guarantee the decidability of WeakNext we have to restrict our target to the set of BPMN processes whose transition system is finitely observable w.r.t. the set of observable labels. Hereafter, we say that a BPMN process p is *well-founded* w.r.t. a set of labels M if

$LTS(s)$ is finitely observable w.r.t. M, where s is the COWS encoding of p. Intuitively, a BPMN process p is well-founded if every cycle in p has at least one activity which is observable. Given the definition of observable labels $\overline{L}$ in Section 3.5, a BPMN process is therefore well-founded if every sequence flow path ending in a cycle contains at least a task or an event handling errors. Restricting the analysis to well-founded processes does not impose a serious limitation in practice. It avoids those degenerate cases where the process could get stuck because no task or event handler is performed but the process is not ended. An example is a BPMN process with a cycle formed only by gates. Note that non well-founded processes can be detected directly on the diagram describing the process.

From the above considerations, we can state the following corollary.

Corollary 1. *Let p be a well-founded BPMN process, s the COWS service encoding p, $LTS(s) = (s, S, L, \rightarrow)$, and $\overline{L} \subseteq L$ the set of observable labels for s. Then,* WeakNext *terminates for all $s' \in S$.*

We now prove that Algorithm 1 terminates for every COWS service s encoding a well-founded BPMN process. If we consider an audit trail l of length k, Algorithm 1 explores only a finite portion of the transition system of s to verify if an entry e of l is accepted, because of Corollary 1. The idea is that, being k finite, Algorithm 1 explores a finite portion of $LTS(s)$, and therefore terminates.

Theorem 1. *Let p be a well-founded BPMN process, s the COWS service encoding p, $\overline{L}$ the set of observable labels for s, and l an audit trail of length k. Then, Algorithm 1 on (s, l) terminates.*

The following result demonstrates the correctness of Algorithm 1.

Theorem 2. *Let s be a COWS service encoding a well-founded BPMN process and l an audit trail. Algorithm 1 on (s, l) returns $\mathtt{true}$ iff there exists a trace $\sigma \in \Sigma(LTS(s))$ such that σ accepts l.*

The results presented in this section allow us to conclude that, within the boundaries defined by a well-founded BPMN process, Algorithm 1 can always decide if a finite audit trail raises concerns about infringement of purpose specification.

6 Related Work

It is largely recognized that traditional access control is insufficient to cope with privacy issues [1,2,22]. Karjoth et al. [23] identify the need of three additional elements (i.e., *purpose*, *condition*, and *obligation*), beside the basic authorization elements (i.e., subject, object, and action). Based on this observation, a number of models, languages and standards tailored to specify data protection policies have been proposed in the last decade [2,4,5,8,24,25,26]. In this section, we discuss existing proposals based on the concept of purpose. Works on obligations are complementary to our work as obligations are intended to address different data protection requirements (e.g., retention period).

Existing purpose-based frameworks [2,4,5,24] treat the intended purpose as a label attached to data. Upon receiving an access request, they match the access purpose against the label attached to the requested data. However, they rely on the fact that the requester specifies the purpose legally, implying complete trust on the requester. Few researchers have addressed the problem of validating the access purpose. For instance, Byun and Li [4] specify the roles that can make a request for a given purpose. However, this approach is preventive and does not solve the problem of re-purposing the data. In contrast, we propose to specify the operational behavior of purposes, which makes it possible to analyze the actual usage of data with respect to purpose specifications.

To best of our knowledge, there is only another framework, the Chain method [27], that attempts to define an operational model for purposes. Here, a privacy policy specifies the “chains of acts” that users are allowed to perform where each act is some form of information handling (i.e., creating, collecting, processing, disclosing); purposes are implicitly defined by the sequences of acts on personal information. Compared to our approach, this corresponds to specifying business processes in terms of actions, introducing an undesirable complexity into process models. Conversely, our solution provides a flexible way to align business processes, data protection policies and audit trails and allows an organization to reuse its business process models for purpose control. In addition, the Chain method has a preventive nature and lacks capability to reconstruct the sequence of acts (when chains are executed concurrently).

Some approaches propose methods for a-posteriori policy compliance [28,29]. For instance, Cederquist et al. [29] present a framework that allows users to justify their actions. However, these frameworks can only deal with a limited range of access control policies and do not consider the purpose. Agrawal et al. [30] propose an auditing framework to verify whether a database system is compliant with data protection policies. However, their focus is mainly on minimizing the information to be disclosed and identifying suspicious queries rather than verifying data usage.

Techniques for detecting system behaviors that do not conform to an expected model have been proposed in process mining and intrusion detection. In intrusion detection, logs of system calls are analyzed either to detect deviations from normal behavior (anomaly detection) [31] or to identify precise sequences of events that damage the system (misuse detection) [32]. Accordingly, our method can be seen as an anomaly detection technique. Process mining [33] and, in particular, conformance checking [13] have been proposed to quantify the “fit” between an audit trail and a business process model. These techniques, however, work with logs in which events only refer to activities specified in the business process model. Consequently, they are not able to analyze the compliance with fine-grained data protection policies. Moreover, they are often based on Petri Nets. This formalism does not make it possible to capture the full complexity of business process modeling languages such as BPMN. Existing solutions based on Petri Nets either impose some restrictions on the syntax of BPMN (e.g., avoiding cycles), or define a formal semantics that deviate from the informal one. Conversely, we have adopted COWS [10] for the representation of organizational processes. This language has been proved to be suitable for representing a large set of BPMN constructs and analyzing business processes quantitatively [16].

7 Conclusions

Organizations often make use of business process models to define how organizational goals should be achieved. In this paper, we proposed to associate the purpose defined in data protection policies to such process models. This representation makes it possible to verify the compliance of the actual data usage with purpose specifications by determining whether the audit trail represents a valid execution of the processes defined by the organization to achieve a certain purpose. We expect that the audit process is tractable and scales to real applications. Intuitively, the complexity of the approach is bound to the complexity of Algorithm 1. Indeed, Algorithm 1 is independent from the particular object under investigation so that it is not necessary to repeat the analysis of same process instance for different objects. In addition, the analysis of process instances is independent from each other, allowing for massive parallelization. A detailed complexity analysis of the algorithm is left for future work, but our first experiments show encouraging performances.

The work presented in this paper suggests some interesting directions for the future work. The proposed approach alone may not be sufficient when we consider the human component in organizational processes. Process specifications may contain human activities that cannot be logged by the IT system (e.g., a physician discussing patient data over the phone for second opinion). These silent activities make it not possible to determine if an audit trail corresponds to a valid execution of the organization process. Therefore, we need a method for analyzing user behavior and the purpose of data usage when audit trails are partial. In addition, many application domains like healthcare require dealing with exceptions. For instance, a physician can take actions that diverge from the procedures defined by a hospital to face emergency situations. On one side, preventing such actions may be critical for the life of patients. On the other side, checking every occurrence of emergency situations can be costly and time consuming. To narrow down the number of situations to be investigated, we are complementing the presented mechanism with metrics for measuring the severity of privacy infringements.

References

1. Guarda, P., Zannone, N.: Towards the Development of Privacy-Aware Systems. Information and Software Technology 51(2), 337–350 (2009)
2. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic Databases. In: Proceedings of the 28th International Conference on Very Large Data Bases, pp. 143–154. Morgan Kaufmann, San Francisco (2002)
3. Ashley, P., Hada, S., Karjoth, G., Schunter, M.: E-P3P privacy policies and privacy authorization. In: Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society, pp. 103–109. ACM, New York (2002)
4. Byun, J.-W., Li, N.: Purpose based access control for privacy protection in relational database systems. VLDB J 17(4), 603–619 (2008)
5. Massacci, F., Mylopoulos, J., Zannone, N.: Hierarchical Hippocratic Databases with Minimal Disclosure for Virtual Organizations. VLDB J 15(4), 370–387 (2006)
6. Catteddu, D., Hogben, G.: Cloud Computing – Benefits, risks and recommendations for information security. European Network and Information Security Agency (ENISA), Report (2009)

7. Daskala, B.: Being diabetic in 2011 – Identifying Emerging and Future Risks in Remote Health Monitoring and Treatment. European Network and Information Security Agency (ENISA), Report (2009)
8. Karjoth, G., Schunter, M., Waidner, M.: Platform for Enterprise Privacy Practices: Privacy-enabled Management of Customer Data. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 69–84. Springer, Heidelberg (2003)
9. Lovis, C., Spahni, S., Cassoni, N., Geissbuhler, A.: Comprehensive management of the access to the electronic patient record: Towards trans-institutional networks. Int. J. of Medical Informatics 76(5-6), 466–470 (2007)
10. Lapadula, A., Pugliese, R., Tiezzi, F.: Calculus for Orchestration of Web Services. In: De Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 33–47. Springer, Heidelberg (2007)
11. OASIS, Web Services Business Process Execution Language – Version 2.0, OASIS Standard (2007), http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html
12. Object Management Group, Business Process Modeling Notation (BPMN) Specification (version 1.2), OMG document (2009), http://www.omg.org/spec/BPMN/1.2/
13. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. 33(1), 64–95 (2008)
14. Enterprise Security and Privacy Authorization (XSPA) Profile of XACML v2.0 for Healthcare, Committee Draft (2008), http://xml.coverpages.org/xspa-xacml-profile-CD01-29664.pdf
15. Plotkin, G.: The origins of structural operational semantics. J. Log. Algebr. Program 60, 3–15 (2004)
16. Prandi, D., Quaglia, P., Zannone, N.: Formal analysis of BPMN via a translation into COWS. In: Wang, A.H., Tennenholtz, M. (eds.) COORDINATION 2008. LNCS, vol. 5052, pp. 249–263. Springer, Heidelberg (2008)
17. Office of the National Coordinator for Health Information Technology Electronic Health Records and Meaningful Use (2010), http://healthit.hhs.gov/portal/server.pt/community/healthit_hhs_gov__meaningful_use_announcement/2996
18. Ma, D., Tsudik, G.: A new approach to secure logging. ACM Trans. Storage 5(1), 1–21 (2009)
19. Schneier, B., Kelsey, J.: Secure audit logs to support computer forensics. ACM Trans. Inf. Syst. Secur. 2(2), 159–176 (1999)
20. Rostad, L., Edsberg, O.: A study of access control requirements for healthcare systems based on audit trails from access logs. In: Proceedings of the 22nd Annual Computer Security Applications Conference, pp. 175–186. IEEE Computer Society, Los Alamitos (2006)
21. Fantechi, A., Gnesi, S., Lapadula, A., Mazzanti, F., Pugliese, R., Tiezzi, F.: A model checking approach for verifying COWS specifications. In: Fiadeiro, J.L., Inverardi, P. (eds.) FASE 2008. LNCS, vol. 4961, pp. 230–245. Springer, Heidelberg (2008)
22. He, Q., Antón, A.I.: A Framework for Modeling Privacy Requirements in Role Engineering. In: Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality, pp. 137–146 (2003)
23. Karjoth, G., Schunter, M.: A Privacy Policy Model for Enterprises. In: Proceedings of the 15th IEEE Workshop on Computer Security Foundations, pp. 271–281. IEEE Computer Society, Los Alamitos (2002)

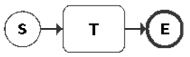
24. Backes, M., Karjoth, G., Bagga, W., Schunter, M.: Efficient comparison of enterprise privacy policies. In: Proceedings of the 2004 ACM Symposium on Applied Computing, pp. 375–382. ACM, New York (2004)
25. Hilty, M., Basin, D.A., Pretschner, A.: On Obligations. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 98–117. Springer, Heidelberg (2005)
26. OASIS, eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard (2005), http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
27. Al-Fedaghi, S.S.: Beyond purpose-based privacy access control. In: Proceedings of the 8th Conference on Australasian Database, pp. 23–32. Australian Computer Society, Inc. (2007)
28. Fournet, C., Guts, N., Nardelli, F.Z.: A formal implementation of value commitment. In: Gairing, M. (ed.) ESOP 2008. LNCS, vol. 4960, pp. 383–397. Springer, Heidelberg (2008)
29. Cederquist, J.G., Corin, R.J., Dekker, M.A.C., Etalle, S., den Hartog, J.I., Lenzini, G.: Audit-based compliance control. Int. J. Inf. Sec. 6(2-3), 133–151 (2007)
30. Agrawal, R., Bayardo, R., Faloutsos, C., Kiernan, J., Rantzau, R., Srikant, R.: Auditing Compliance with a Hippocratic Database. In: Proceedings of the 30th International Conference on Very Large Data Bases. VLDB Endowment, pp. 516–527 (2004)
31. Feng, H.H., Kolesnikov, O.M., Fogla, P., Lee, W., Gong, W.: Anomaly detection using call stack information. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 62–75. IEEE Computer Society, Los Alamitos (2003)
32. Kumar, S., Spafford, E.H.: A Pattern Matching Model for Misuse Intrusion Detection. In: Proceedings of the 17th National Computer Security Conference, pp. 11–21 (1994)
33. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Trans. Knowl. Data Eng. 16(9), 1128–1142 (2004)

A Encoding of BPMN in COWS

In this section we present some examples that provide the intuition underlying the COWS semantics and the encoding of BPMN processes into COWS specification.

Consider the simple process in Fig. 7(a); it is composed by a start event S, a task T, and an end event E within a pool P. The corresponding COWS service is $Serv = [[S]] \mid [[T]] \mid [[E]]$, where services $[[S]]$, $[[T]]$, and $[[E]]$ are defined in Fig. 7(b). Service $[[S]]$ gives the control to task T in pool P, written as $P \cdot T!\,\langle\rangle$. Service $[[T]]$ receives the control ($P \cdot T?\,\langle\rangle$) and then (represented as infix dot "$.$") gives the control to the end event E within pool P ($P \cdot E!\,\langle\rangle$). Finally, $[[E]]$ closes the flow receiving the control $P \cdot E?\,\langle\rangle$. The LTS associated with service $Serv$ (Fig. 7(c)) gives a compact representation of the possible paths of tokens within the process of Fig. 7(a). In this simple case, only a single path is possible.

An example involving a gateway is presented in Fig. 8(a). Here, when reaching the exclusive gateway G, the token can follow only one flow, either through $T1$ or through $T2$. Fig. 8(b) shows the encoding of the process in COWS. Note that the encoding of G, $[[G]]$, makes use of $\mathbf{kill}(k)$: when an alternative is selected, a killer signal is sent to prevent the other alternative to be executed. This is evident in Fig. 8(c) where state $St6$ is reached by running either $T1$ or $T2$, but there is no path where both $T1$ and $T2$



(a) BPMN Process

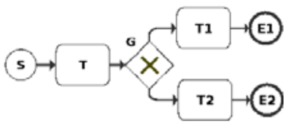
$$
\begin{aligned}
&[[S]] := P \cdot T!\,\langle\rangle\\
&[[T]] := P \cdot T?\,\langle\rangle.\,P \cdot E!\,\langle\rangle\\
&[[E]] := P \cdot E?\,\langle\rangle
\end{aligned}
$$

(b) COWS *Serv*

$$\begin{tikzcd} St1 \arrow[r, "P\cdot T"] & St2 \arrow[r, "P\cdot E"] & St3 \end{tikzcd}$$

(c) COWS LTS

Fig. 7. A Simple BPMN process in COWS



(a) BPMN Process

$$
\begin{aligned}
&[[S]] := P \cdot T!\,\langle\rangle\\
&[[T]] := P \cdot T?\,\langle\rangle.\,P \cdot G!\,\langle\rangle\\
&[[G]] := P \cdot G?\,\langle\rangle.\,[\,k\,][\,sys\,](\\
&\quad sys \cdot T1!\,\langle\rangle \mid sys \cdot T2!\,\langle\rangle \mid\\
&\quad sys \cdot T1?\,\langle\rangle.(\mathbf{kill}(k) \mid \{\!|P \cdot T1!\,\langle\rangle|\!\}) \mid\\
&\quad sys \cdot T2?\,\langle\rangle.(\mathbf{kill}(k) \mid \{\!|P \cdot T2!\,\langle\rangle|\!\}))\\
&[[T1]] := P \cdot T1?\,\langle\rangle.\,P \cdot E1!\,\langle\rangle\\
&[[E1]] := P \cdot E1?\,\langle\rangle\\
&[[T2]] := P \cdot T2?\,\langle\rangle.\,P \cdot E2!\,\langle\rangle\\
&[[E2]] := P \cdot E2?\,\langle\rangle
\end{aligned}
$$

(b) COWS *Serv*

$$\begin{tikzcd}
 & St1 \arrow[d, "P\cdot T"] & \\
 & St2 \arrow[d, "P\cdot G"] & \\
 & St3 \arrow[dl, "sys\cdot T1"'] \arrow[dr, "sys\cdot T2"] & \\
St4 \arrow[d, "P\cdot T1"'] & & St7 \arrow[d, "P\cdot T2"] \\
St5 \arrow[dr, "P\cdot E1"'] & & St8 \arrow[dl, "P\cdot E2"] \\
 & St6 &
\end{tikzcd}$$

(c) COWS LTS

Fig. 8. A BPMN Process with Exclusive Gateway

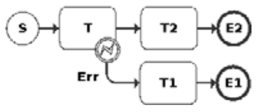
are executed. We use the private name sys to avoid interference between services or between different executions of the same service.

Tasks with an associated error event are another example of sequence flow. The example in Fig. 9(a) models a task T that can either proceed "correctly" to task $T2$ or incur an error *Err* managed by task $T1$. The encoding of the COWS services is reported in Fig. 9(b). Note that the encoding of the BPMN elements S, $T1$, $E1$, $T2$, and $E2$ are the same of the previous example (Fig. 8(b)). We only change the definition of $[[T]]$: when T takes the token, it can either proceed normally invoking $T2$ ($P \cdot T2!\,\langle\rangle$) or after signaling an error (label "$sys \cdot Err$") proceed invoking $T1$ ($P \cdot T1!\,\langle\rangle$). The LTS of Fig. 9(c) shows these two possible paths.

The three examples above give some hints about the representation of an organizational process through COWS. To have a complete insight in the COWS services used to model the process in Fig. 1 we have to discuss two further topics, namely *message flows* and *service replication*.

Message flows are represented as communications over endpoints belonging to different pools. An example is given in Fig. 10. Here, events $S2$ and $S3$ are message start events, i.e. events that trigger the start of the process upon the receipt of a message, and $E1$ and $E2$ are message end events, i.e. events that send a message to a participant at the conclusion of the process. When the end event $E1$ is triggered, it sends a message *msg1* to the start event $S3$ in pool $P2$; similarly, end event $E2$ sends a message *msg2* to the start event $S2$ in pool $P1$. Upon the receipt of the message, $S2$ and $S3$ start the corresponding process. The COWS encoding of the process in Fig. 10(a) is presented in Fig. 10(b) and the corresponding LTS in Fig. 10(c).

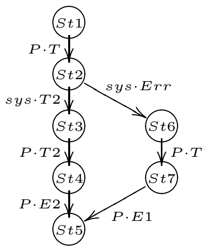
In general, it is not known in advance how many times a service is invoked during the execution of a process. An example of this is given by cycles. Cycles are closed paths in the process; they can consist of sequence flows, as the cycle involving $T01$, $G1$,



(a) BPMN Process

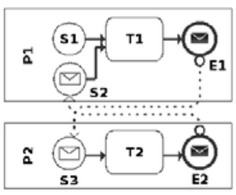
$$
\begin{aligned}
&[[S]] := P \cdot T!\,\langle\rangle\\
&[[T]] := P \cdot G?\,\langle\rangle.\,[\,k\,][\,sys\,](\\
&\quad sys \cdot Err!\,\langle\rangle \mid sys \cdot T2!\,\langle\rangle \mid\\
&\quad sys \cdot Err?\,\langle\rangle.(\mathbf{kill}(k) \mid \{\!|P \cdot T1!\,\langle\rangle|\!\}) \mid\\
&\quad sys \cdot T2?\,\langle\rangle.(\mathbf{kill}(k) \mid \{\!|P \cdot T2!\,\langle\rangle|\!\}))\\
&[[T1]] := P \cdot T1?\,\langle\rangle.\,P \cdot E1!\,\langle\rangle\\
&[[E1]] := P \cdot E1?\,\langle\rangle\\
&[[T2]] := P \cdot T2?\,\langle\rangle.\,P \cdot E2!\,\langle\rangle\\
&[[E2]] := P \cdot E2?\,\langle\rangle
\end{aligned}
$$

(b) COWS *Serv*



(c) COWS LTS

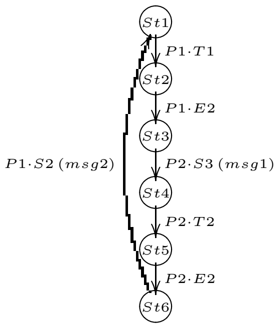
Fig. 9. A BPMN Process with Error Event



(a) BPMN Process

$$
\begin{aligned}
&[[S1]] := P1 \cdot T1!\,\langle\rangle\\
&[[S2]] := *\,[\,z\,]P1 \cdot S2?\,\langle z\rangle.\,P1 \cdot T1!\,\langle\rangle\\
&[[T1]] := *\,P1 \cdot T1?\,\langle\rangle.\,P1 \cdot E1!\,\langle\rangle\\
&[[E1]] := *\,P1 \cdot E1?\,\langle\rangle.\,P2 \cdot S3!\,\langle msg1\rangle\\
&[[S3]] := *\,[\,z\,]P2 \cdot S3?\,\langle z\rangle.\,P2 \cdot T2!\,\langle\rangle\\
&[[T2]] := *\,P2 \cdot T2?\,\langle\rangle.\,P2 \cdot E2!\,\langle\rangle\\
&[[E2]] := *\,P \cdot E2?\,\langle\rangle.\,P1 \cdot S2!\,\langle msg2\rangle
\end{aligned}
$$

(b) COWS *Serv*



(c) COWS LTS

Fig. 10. A BPMN Process with Message Flow and Cycles

$T02$, and again $T01$ in Fig. 1, or combination of sequence and message flows, as the cycle involving $S2$, $T1$, $E1$, $S3$, $T2$, $E2$, and again $S2$ in Fig. 10(a). A cycle involves the restart of the process from a certain activity. Consider the example of Fig. 1: if the GP is not able to make a diagnosis (*Err* in task $T02$), the process has to restart from $T01$. To address this issue, we prefix COWS services with the replication operator $*$ (Fig. 10(b)). This operator makes multiple copies of the COWS service; each copy corresponds to an invocation of the service.

B Proofs

Proof of Proposition 1. If $LTS(s)$ is finitely observable w.r.t. $\overline{L}$, then for each trace

$$s \xrightarrow{l_0} s_0 \xrightarrow{l_1} s_1 \ldots s_n \xrightarrow{l_{n+1}} s_{n+1} \ldots$$

there exists a value $k < \infty$ such that $l_k \in \overline{L}$, by Def. 8. This implies that the set of states that can be reached from s with exactly one label in $\overline{L}$ can be computed in a finite number of steps, namely $\mathsf{WeakNext}(s)$ is decidable on $\overline{L}$. $\square$

Proof of Theorem 1. The proof is by induction on the length k of $l = e_1 e_2 \dots e_k$.

Base Step: Let $l = e_1$. A BPMN process is always triggered by a start event [12]. Therefore, the initial configuration has the form $conf = (s, empty, \mathsf{WeakNext}(s))$. By definition $LTS(s)$ is a finitely observable LTS. Therefore, each trace $\sigma \equiv (s, l_0), (s_0, l_1) \dots (s_n, l_n) \dots$ in $\Sigma(LTS(s))$ is such that $\exists j < \infty . l_j \in \overline{L}$. If $\Sigma(LTS(s))$ is empty, $\mathsf{WeakNext}(s)$ returns an empty set. In this case, the algorithm does not enter into the forall of line 7 and the variable *found* remains false. Thereby, the algorithm exits at line 22 with *false*. If there exists $(l_j, s', active\_tasks) \in \mathsf{WeakNext}(s)$ such that $l_j = r \cdot e_1.task$ with $e_1.role \leq_R r$, or $l_j = sys \cdot Err$ and $e_1.status = failure$, Algorithm 1 returns *true* at line 24. Otherwise, Algorithm 1 returns *false* at line 22.

Inductive Step: Let $l = e_1 \dots e_k$ with $k > 1$. By the inductive hypothesis, Algorithm 1 terminates on the audit trail $l^{(k-1)} = e_1 e_2 \dots e_{k-1}$. Let $conf\_set$ be the set of actual configurations. If $conf\_set = \emptyset$, the variable *found* remains *false*, and Algorithm 1 returns *false* at line 22. If there exists a configuration $conf \in conf\_set$ such that $(r, e_k.task) \in conf.active\_tasks$ with $e_k.role \leq_R r$ and $e_k.status = success$, the configuration is added to the set of configurations to be considered in the next iteration (line 16). As l is completely analyzed and *found* is equal to *true* (line 15), Algorithm 1 returns $true$ at line 24. If there exists a configuration $conf \in conf\_set$ such that $(r \cdot e_k.task, s', at) \in conf.next$ with $e_k.role \leq_R r$ or $(sys \cdot Err, s', at) \in conf.next$ and $e_k.status = failure$, variable $found$ becomes $true$ at line 11. Then, Algorithm 1 returns $true$ at line 24. Otherwise, if the entry e_k does not correspond either to any task in $conf.active\_tasks$ or to any label in $conf.next$, Algorithm 1 returns $false$ on line 22. Therefore, by induction, Algorithm 1 terminates for l. □

Proof of Theorem 2. The correctness is essentially given in term of soundness (forwards proof) and completeness (backwards proof). We show the implications separately. We only sketch the proof of the theorem, which requires a double induction on the length of l and on the structure of s.

($\Longrightarrow$) Let $l = e_1 e_2 \dots e_k$ be of length k. Algorithm 1 on (s,l) returns *true* only if the while cycle of line 3 ends and line 24 is executed. By Theorem 1, we know that the algorithm and, consequently, the cycle always terminates. Line 24 is executed only if for each $e_i \in l$ either condition on line 8 is not verified (i.e., $e_i.task$ is active and $e_i.status$ is not *failure*) or, by line 10, there exists a configuration $conf \in conf\_set$ that accepts e_i (i.e., $(r \cdot e_i.task, s', at) \in conf.next$ with $e_i.role \leq_R r$ or $(sys \cdot Err, s', at) \in conf.next$ and $e_k.status = failure$). This consideration makes it possible to prove that, at each iteration $i \in [1, k]$ of the while cycle, Algorithm 1 computes the set of traces of $LTS(s)$ that accept the prefix $e_1 \dots e_i$. Therefore, if line 24 is executed, there exists at least one trace σ in $\Sigma(LTS(s))$ that accepts l.

($\Longleftarrow$) To prove the completeness of Algorithm 1, we have to prove that if there is a trace from s that accepts l, then Algorithm 1 on (s, l) gives *true*. Below, we demonstrate the contra-positive of the previous sentence, i.e. if Algorithm 1 on (s, l) returns *false*, then there is not a trace from s that accepts l. Given $l = e_1 e_2 \dots e_k$, Algorithm 1 on (s, l) returns *false* if there exists an iteration $i \in [1, k]$ of the while cycle in line 3 such that the condition on line 21 is true. This is possible if during iteration i both line 11 and line 15 are not executed. The first condition is verified if, given a $conf \in conf\_set$ such that e_i does not correspond to any task in $conf.active\_tasks$ or it is a failure (i.e., condition on line 8 is true), there is not a triple $(l_m, s_m, at) \in conf.next$ that accepts e_i (condition on line 10 is false). In this case, by Proposition 1, there does not exist a finitely observable trace $(s_j, l_{j+1}) \ldots (s_{m-1}, l_m)$ from the current state s_j to s_m such that e_i corresponds to l_m. Line 15 is executed only if e_i corresponds to a task in $conf.active\_tasks$ and it is not a failure (i.e., condition on line 8 is false). No execution of this line implies that there does not exist a $conf \in conf\_set$ where e_1 is active. This means that e_i does not correspond to an active task in the current state s_j. Suppose that at the i-th iteration of the while cycle, Algorithm 1 has already built all the traces accepting $e_1 \ldots e_{i-1}$. If Algorithm 1 cannot replay e_i in the process, we can conclude that there is not a trace in $\Sigma(LTS(s))$ accepting $e_1 \ldots e_i$, and so l. $\square$

Author Index