LNCS 4721

Willem Jonker
Milan Petković (Eds.)

Secure Data Management

4th VLDB Workshop, SDM 2007
Vienna, Austria, September 2007
Proceedings

Springer

Willem Jonker   Milan Petković (Eds.)

Secure Data Management

4th VLDB Workshop, SDM 2007
Vienna, Austria, September 23-24, 2007
Proceedings

Springer

Volume Editors

Willem Jonker
Milan Petković
Philips Research Europe
High Tech Campus 34
5656 AE Eindhoven, The Netherlands
E-mail: {willem.jonker, milan.petkovic} @philips.com

Library of Congress Control Number: 2007935172

CR Subject Classification (1998): H.2.0, H.2, C.2.0, H.3, E.3, D.4.6, K.6.5

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

ISSN	0302-9743
ISBN-10	3-540-75247-1 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-75247-9 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12165858 06/3180 5 4 3 2 1 0

Preface

Although a number of cryptography and security techniques have been around for quite some time, emerging technologies, such as ubiquitous computing and ambient intelligence that exploit increasingly interconnected networks, mobility and personalization, put new requirements on privacy and security with respect to data management. As data are accessible anytime anywhere, according to these new concepts, it becomes much easier to get unauthorized data access. As another consequence, the use of new technologies has brought some privacy concerns. It becomes simpler to collect, store, and search personal information and endanger people's privacy. Therefore, research in the area of secure data management is of growing importance, attracting the attention of both the data management and security research communities. The interesting problems range from traditional topics, such as access control and general database security, via privacy preserving data mining, to new research directions, such as search on encrypted data and privacy-enhancing technologies.

This year, the call for papers attracted 29 papers both from universities and industry. For presentation at the workshop, the Program Committee selected 11 full papers (37% acceptance rate) as well as 4 position papers. These papers are also collected in this volume, which we hope will serve you as a useful research and reference material.

The papers in this volume are divided into four major sections. The first section focuses on access control, which remains an important area. The papers in this section address conflict resolution, administrative policies, and audit of access rights. The second section changes slightly the focal point to the more general topics of database security. The papers in this section deal with disclosure control, authentication of query results, intrusion detection, search on encrypted data, and XML security. The third section focuses on privacy protection addressing the topics of k-anonymity, spyware, and privacy management in healthcare. The last section collects four position papers whose topics range from electronic health record requirement analysis to database privacy issues.

We would like to acknowledge Richard Brinkman, who helped in the technical preparation of this proceedings.

July 2007

Willem Jonker
Milan Petković

Organization

Workshop Organizers

Willem Jonker (Philips Research/University of Twente, The Netherlands)
Milan Petković (Philips Research, The Netherlands)

Program Committee

Gerrit Bleumer, Francotyp-Postalia, Germany
Ljiljana Branković, University of Newcastle, Australia
Sabrina De Capitani di Vimercati, University of Milan, Italy
Ernesto Damiani, University of Milan, Italy
Eric Diehl, Thomson Research, France
Lee Dong Hoon, Korea university, Korea
Jeroen Doumen, Twente University, The Netherlands
Jan Eloff, University of Pretoria, South Africa
Csilla Farkas, University of South Carolina, USA
Eduardo Fernández-Medina, University of Castilla-La Mancha, Spain
Elena Ferrari, Università degli Studi dell'Insubria, Italy
Simone Fischer-Hübner, Karlstad University, Sweden
Tyrone Grandison, IBM Almaden Research Center, USA
Dieter Gollmann, Technische Universität Hamburg-Harburg, Germany
Ehud Gudes, Ben-Gurion University, Israel
Hakan Hacigumus, IBM Almaden Research Center, USA
Marit Hansen, Independent Centre for Privacy Protection, Germany
Min-Shiang Hwang, National Chung Hsing University, Taiwan
Mizuho Iwaihara, Kyoto University, Japan
Sushil Jajodia George Mason University, USA
Ton Kalker, HP Research, USA
Marc Langheinrich, Institute for Pervasive Computing ETH Zurich, Switzerland
Nguyen Manh Tho, Vienna University of Technology, Austria
Nick Mankovich, Philips Medical Systems, USA
Sharad Mehrotra, University of California at Irvine, USA
Stig Frode Mjlsnes, Norwegian University of Science and Technology, Norway
Eiji Okamoto, University of Tsukuba, Japan
Sylvia Osborn, University of Western Ontario, Canada
Gnther Pernul, University of Regensburg, Germany
Birgit Pfitzmann, IBM Zurich Research Lab, Switzerland
Bart Preneel, KU Leuven, Belgium
Kai Rannenberg, Goethe University Frankfurt, Germany

Andreas Schaad, SAP Labs, France
Morton Swimmer, IBM Zurich Research Lab, Switzerland
Sheng Zhong, Stevens Institute of Technology, USA

Additional Referees

Ludwig Fuchs, University of Regensburg, Germany
Anna Zych, Twente University, The Netherlands
Lei Zhang, George Mason University, USA

Table of Contents

Positon Papers

A Unified Conflict Resolution Algorithm

Amir H. Chinaei, Hamid R. Chinaei, and Frank Wm. Tompa

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON N2L 3G1, Canada
Tel.: +1 (519) 888-4567 extensions {36612, 33400, 34675}
{ahchinaei,hrchinaei,fwtompa}@uwaterloo.ca

Abstract. While some authorization models support either positive or negative authorizations, hybrid frameworks take advantage of both authorizations. Resolving authorization conflicts is quite a challenge due to the existence of sophisticated inheritance hierarchies and the diversity of ways to combine resolution policies. Some researchers have addressed conflict resolution for tree-structured hierarchies, and others have applied a simple conflict resolution policy. The challenge is to combine several policies and to support sophisticated structures in one single framework. This paper proposes a unified framework together with a single parametric algorithm that supports all the legitimate combinations simultaneously, based on four conflict resolution policies. We validate our approach by testing the algorithm against both real data and synthetic examples to provide extensive experimental results.

Keywords: Access Control Models, Hybrid Authorizations, Conflict Resolution.

1 Introduction

Because individual users can assume several personae (e.g., payroll clerk, member of the social committee, and occupant on the fourth floor), they may be simultaneously authorized for some activity and denied authorization for that same activity when viewed in another role. There exist several conflict resolution policies, such as "denial takes precedence" and "the most specific authorization takes precedence," in the literature of access control models. However, combining the policies provides a variety of different comprehensive conflict resolution strategies, which have not been well addressed. Designers of access control models typically choose a specific approach to conflict resolution and incorporate a hardwired conflict resolution method within their models. Consequently, if an enterprise subsequently decides to choose an alternative conflict resolution strategy, the whole system has to be replaced. Maintaining separate software for multiple strategies is expensive for software providers. We propose one single parameterized algorithm by which security administrators can invoke a chosen strategy, among many, without needing to reinstall the whole system.

1.1 Motivating Example

Figure 1 illustrates a subject inheritance hierarchy including nine subjects. The arrows represent group membership (e.g., subjects S_4 and S_5 are members of subject/group S_3) and the sign labels represent explicit authorizations (+ indicates positive authorization and – represents denial). For simplicity of exposition, we assume that access to an object is either granted or denied (rather than separately controlling reading, writing, and other operators), and we illustrate only authorizations for a single object. The figure shows that subjects S_2 and S_4 are explicitly authorized to access the object, whereas subject S_5 is explicitly denied from accessing it.

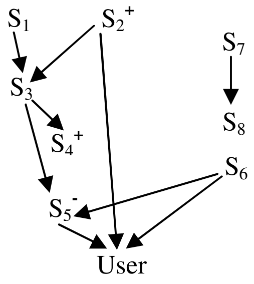


Fig. 1. An example of the subject hierarchy

Given the data in Figure 1, assume we are interested in knowing whether or not subject *User* is authorized to access the object. One may interpret the data to mean that the object is accessible to subject *User* since *User* is a descendant of S_2 and thereby inherits S_2's authorizations. However, another may argue that the object should not be accessible to *User* since he is a member of S_5 which is denied access. In fact, there is a conflict in the system. Conflict resolution policies are needed to answer such questions.

Adopting one simple policy, such as “the most specific authorization takes precedence,” is not sufficient in practice. For instance, such a policy is insufficient where the subject hierarchy is more complex than tree-based structures and therefore, a subject may have more than one “most specific” authorization. For example, in Figure 1, neither S_2 nor S_5 is more specific to *User*, with respect to the other, since both of them are at the minimum distance of 1 from *User*. Furthermore, there are situations in which the highest authority (not the most specific one) should be the final arbiter. For instance, assume a student is authorized by the university athletic office to referee hockey games on campus (which requires more than 20 hours per week for several weeks); however, he is required by the department not to accept heavy non-departmental tasks (in order to comply with his full-time registration status). In such a case, the university administration may resolve the conflict by deciding to let him to referee for a limited time. To visualize such a case, assume there is an edge from S_1 to S_2 in Figure 1 and S_1 is labeled positively. Representing the student by *User*, the referees group by S_2, the members of the department by S_5, and the university members by S_1, it is apparent that for this enterprise the most global authorization should take precedence in resolving the conflict.

Some have proposed the “negative takes precedence” policy, but this too is not universally acceptable. For instance, conflicts often are resolved by the “majority takes precedence” rule in voting systems. Additionally, the open policy recommends a default positive authorization for subjects which are not explicitly permitted to access a particular object [6, 10]. Therefore, there are applications in which “positive takes precedence.”

Even from these simple examples we see that it is required to combine various conflict resolution policies to obtain a comprehensive conflict resolution framework that supports several strategies simultaneously. Moreover, each policy may encompass several variants, and consequently many strategy instances are possible. What are all the legitimate strategy instances? Is there a unified algorithm to support all instances parametrically?

Chinaei and Zhang addressed the first question by providing a conflict resolution framework in which 32 legitimate instances are supported [2]. In that same paper, they proposed the *Dominance()* algorithm for one of the compound conflict resolution strategies and provided some guidelines to extend the algorithm to other strategies. However, designing a single parameterized conflict resolution algorithm to support all combined strategies remained open.

In this paper, we first extend the framework to support 16 additional strategy instances. Thereafter, we present a unified parametric algorithm that supports the comprehensive conflict resolution framework for hybrid authorization models in which the subject hierarchy is a directed acyclic graph.

1.2 Outline

Section 2 reviews four major conflict resolution policies and consequent combined strategies as presented in Chinaei and Zhang’s work [2], and extends the framework to support 48 strategy instances. Section 3 describes our unified parametric algorithm and justifies how the algorithm supports all the instances. Section 4 presents our experimental results. Section 5 reviews the literature. Finally, Section 6 summarizes our contributions and addresses several future directions.

2 Conflict Resolution Models

Before proceeding, we assume the reader is familiar with the following terminology. Access control data can be viewed conceptually as being represented by an access control matrix, where the rows represent subjects, the columns represent objects, and authorizations are stored at the intersections [10]. However, not every subject-object pair has explicit rights assigned. Instead, access control for subject/object pairs with no explicit rights must be derived by other means, e.g. through inheritance via the subject hierarchy.

We therefore distinguish between an *effective access control matrix*, which represents all explicit and derived authorizations, and an *explicit access control matrix*, which represents explicit authorizations only. Given an explicit matrix, conflict resolution strategies are used to fill in all derived authorizations to determine the effective matrix. It is important to note that the explicit matrix is typically very sparse in practice and that the effective matrix is by definition completely filled; therefore practical systems will store the explicit matrix and compute access control authorizations as needed by executing a conflict resolution algorithm on an appropriately extracted subset of that matrix.

Chinaei and Zhang outlined four main policies included in their conflict resolution framework: *preferred authorization*, *locality* (or *globalization*), *majority*, and *default authorization* [2]. These policies have been articulated by other researchers and appear in various real world applications, but they are typically discussed independently and not in combination. In Section 2.1, we restate each policy briefly and independently of other policies, as well as providing some examples of their applicability. Then, in Section 2.2 we explain how legitimate combinations of these policies lead us to define several consequent strategy instances.

2.1 Conflict Resolution Policies

Our model assumes that the subjects for whom authorizations are to be determined are structured as a directed acyclic graph. Individuals are represented as sink nodes; a group of individuals is represented by a node with outgoing edges to each member of the group; a group of groups is represented by a node with outgoing edges to each subgroup member of that group. In general, a group can have zero or more subgroups and zero or more individual nodes. We do not restrict the subject hierarchy to form a tree.

All authorizations of a group may be applicable to a member of that group. As illustrated in Section 1, propagating explicit authorizations to derive effective authorizations of the group members may cause conflicts. Access control must determine for each subject/object/operation whether the subject is to be allowed to execute the operation on the object or denied such execution. Conflict resolution is required when propagating authorizations results in no decision for a particular subject/object/operation triple or when both positive and negative authorizations can be derived for that triple. Here, we outline popular conflict resolution policies that we include in our model.

Preference Policy. Preferred authorization (with one of two modes: either positive or negative) is determined by the system installer at configuration time. This policy determines which authorization wins when both positive and negative authorizations (or neither negative nor positive authorization) can be derived for a particular subject. Negative authorization is preferred (known as closed policy) in more restricted systems such as military; positive authorization may be preferred in more open applications such as public information systems.

Locality Policy. The common mode of this distance-based policy states that the most specific authorization takes precedence. It applies to distributed organizations whose local branches may recognize an exception to a general rule. For instance, a department in a university may admit an outstanding applicant although the general admission requirement is not completely met. Thus, for a given subject, when both positive and negative authorizations can be derived from different ancestors, the one that is closer to the subject wins. Note that the distance between two nodes (subjects) in a directed acyclic graph is measured by computing the shortest directed path.

The locality policy is not deterministic since no authorization wins when the distances are equal.

As an alternative for the locality policy, some enterprises might choose "globality", where the most general authorization takes precedence. One application of this policy is in distributed organizations whose headquarters makes the final decision on a pre-approved task by a local office. For instance, a supreme court may override the appealed decision. For a given subject, when both positive and negative authorizations can be derived from different ancestors, the one that is farther from the subject wins. Similar to the usual locality policy, globality is not deterministic since no authorization may win.

Majority Policy. This policy states that the conflict can be resolved based on votes, and the authorization that has the majority wins. The application of this policy is in situations where several parties have different opinions for giving or not giving the authorization to a particular member and the decision is made by votes. For instance, GATT's current members vote to determine if a new application can get into the group. By applying this policy, the dominant authorization takes precedence. This policy is also non-deterministic since it can result in a tie.

Default Policy. This policy is applied only to root nodes for which no authorization has been defined. Closed systems, such as in the military, require negative authorization by default; however, open systems, such as public information applications, initially allow any subject to enjoy a positive authorization. This policy is deterministic and has two modes (default positive or default negative), but applies to root nodes only. Note that for non-root nodes only the preference policy is deterministic.

2.2 Combined Strategies

Figure 2 illustrates five conflict resolution strategies presented in Chinaei and Zhang's work [2], based on combining the popular conflict resolution policies summarized in Section 2.1. They are given the mnemonics DLP, DLMP, DP, DMLP, and DMP, in which D, L, M, and P indicate Default, Locality, Majority, and Preference policies, respectively. Two properties are guaranteed: first, none of the policies are redundant, and second, there is no conflict after applying the last step. Note that in this framework the Default and Preference policies are always the first and the last applicable policies, respectively, and the other two policies, Locality and Majority, are optional.

Chinaei and Zhang state that because the Default, Locality, and Preference policies can take two modes each, there are 32 different strategies instances in total that can be derived from Figure 2 [2]. (Paths ending with *a*, *b*, and *d* generate eight instances each, and paths ending with *c* and *e* generate four instances each.)

We recognize that there are also applications in which the default policy is not appropriate. For instance, in determining whether S_3 is authorized access according to Figure 1, we may wish to give priority to the explicit authorization on S_2 regardless of whether the value assigned to S_1 is positive or negative. This can be accomplished in general by omitting the default policy, using locality or majority as desired to arbitrate a value for S_3 and using the preferred authorization to assign a value to S_1 only afterwards.

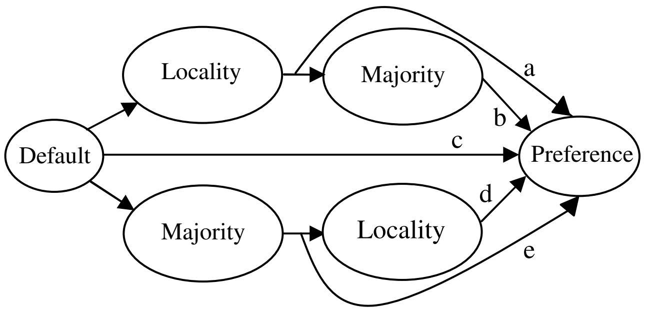


Fig 2. Combined conflict resolution strategies

Therefore, to achieve a comprehensive framework, we augment the combined strategies illustrated in Figure 2 with making the default policy optional as well. This results in five more combined strategies namely LP, LMP, P, MLP, and MP. Hence, the framework supports 16 more instances; strategies LP, LMP, and MLP generate four instances each, and strategies P and MP generate two instances each.

Note that no other combined strategy can be meaningfully composed from these basic conflict resolution policies. For example, the preference policy cannot be optional and must be considered last, since it is the only policy that is well-defined on every node.

3 Implementation

This section describes an algorithm that propagates explicit authorizations through the subject hierarchy, and resolves the possible conflicts based on any of the 48 strategy instances illustrated in Section 2. In particular, Section 3.1 describes details of our conflict resolution algorithm (called *Resolve()*). After that, Section 3.2 describes the propagation of explicit authorizations.

To determine whether a given object, o_j, is effectively accessible to a given subject, S_i, with respect to a given right, r_k, the idea is to apply the following four-step procedure:

Step 1: Consider the maximal sub-graph (called *H*) of the subject hierarchy in which S_i is the sole sink and all other nodes are its ancestors.

Step 2: Assign a letter “d” to all root subjects in *H* that are unlabeled with respect to object o_j and right r_k.

Figure 3 illustrates the result of Steps 1 and 2 for subject *User*, object *obj*, and right *read*, illustrated in Figure 1 as the motivating example.

Step 3: Propagate all authorization labels down every path to subject *User* and store the distance of each propagated authorization from its source node to its destination node (*User*). For instance, the distance of label - (on S_5) to node *User* is 1; also, there are two distances for label “d” (on S_6) to node *User*: one (with value 1) directly from S_6 to *User*, and one (with value 2) via S_5.

Table 1 illustrates the result of authorization propagation for subject *User*, object *obj*, and right *read* as represented by Figure 3.

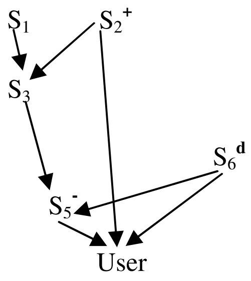


Fig. 3. Sub-graph of subject *User*

Table 1. All *read* authorizations of *User* on *obj*

subject	**object**	**right**	**dis**	**mode**
User	obj	read	1	-
User	obj	read	1	d
User	obj	read	2	d
User	obj	read	1	+
User	obj	read	3	+
User	obj	read	3	d

Table 2. Resolved authorization for each combined strategy

strategy	mode	strategy	mode	strategy	mode	strategy	mode
D$^+$L**M**P$^+$	+	D$^+$L**P**$^+$	+	LM**P**$^+$	+	D$^+$**M**LP$^+$	+
D$^+$L**M**P$^-$	+	D$^+$L**P**$^-$	-	LM**P**$^-$	-	D$^+$**M**LP$^-$	+
D$^-$L**M**P$^+$	-	D$^-$L**P**$^+$	+	**G**MP$^+$	+	D$^-$**M**LP$^+$	-
D$^-$L**M**P$^-$	-	D$^-$L**P**$^-$	-	**G**MP$^-$	+	D$^-$**M**LP$^-$	-
D$^+$**G**MP$^+$	+	D$^+$**G**P$^+$	+	**M**P$^+$	+	D$^+$**M**GP$^+$	+
D$^+$**G**MP$^-$	+	D$^+$**G**P$^-$	+	**M**P$^-$	+	D$^+$**M**GP$^-$	+
D$^-$GM**P**$^+$	+	D$^-$G**P**$^+$	+	L**P**$^+$	+	D$^-$**M**GP$^+$	-
D$^-$GM**P**$^-$	-	D$^-$G**P**$^-$	-	L**P**$^-$	-	D$^-$**M**GP$^-$	-
D$^+$**M**P$^+$	+	D$^+$**P**$^+$	+	**G**P$^+$	+	**M**LP$^+$	+
D$^+$**M**P$^-$	+	D$^+$**P**$^-$	-	**G**P$^-$	+	**M**LP$^-$	+
D$^-$**M**P$^+$	-	D$^-$**P**$^+$	+	**P**$^+$	+	**M**GP$^+$	+
D$^-$**M**P$^-$	-	D$^-$**P**$^-$	-	**P**$^-$	-	**M**GP$^-$	+

Step 4: Apply a particular conflict resolution strategy to resolve any conflicts and derive a final *effective* authorization for the triple $<S_i, o_j, r_k>$.

Table 2 illustrates the result of applying each of the 48 strategy instances (explained in Section 2) to Table 1. For example, D$^+$LMP$^+$ is the strategy instance in which first the default policy is applied and every root subject which is null is initialized to +; then, if there is a conflict, the Locality policy ("the most specific authorization takes precedence") is applied; then if there is still a conflict, the Majority policy is applied; and finally, if the conflict is not resolved, the preference policy in which the positive (+) authorization takes precedence is applied. Let's see what the result of this strategy instance is on Table 1: by applying the default policy D$^+$, all mode d's are replaced by +; by applying the locality policy, the conflict is not resolved since there are conflicting modes + and - from the shortest distance 1; however, by applying the majority rule, mode + wins over mode - since there are more + entries than - entries. Note that the preference policy is not applicable to this case since the conflict is resolved before this rule is triggered; however in other hierarchies the conflict need not yet have been resolved.

For each strategy instance in Table 2, we use a bold font to show which policy has determined the effective authorization when applied to our example. For example, in the last strategy instance, MGP$^-$, by applying the first policy (Majority), the positive authorization wins since there are two +'s (rows 4 and 5) as opposed to only one – (row 1) in Table 1. Therefore, the localization and preference policies of the MGP$^-$ instance are not applicable to this case.

3.1 Algorithm *Resolve()*

This **section** defines our conflict resolution algorithm. Figure 4 illustrates Algorithm *Resolve()* which computes the derived authorization mode of a given subject with respect to a given object and right. The algorithm parameters are *s*, *o*, *r*, *dRule*, *lRule*, *mRule*, and *pRule;* and the result is either + or -. Parameters *s*, *o*, and *r* designate a particular subject, object, and right, respectively, on which the caller is interested to know whether or not the object is accessible to the subject with respect to the specified right. Parameters *dRule*, *lRule*, *mRule*, and *pRule* determine the conflict resolution strategy based on which the final right of the subject on the object must be derived. In particular, parameter *dRule* represents the default policy and takes either of the three values "+", "-", or "0", which respectively states that the unlabelled root ancestors of the subject are to be initialized to positive authorization, negative authorization, or remain null (no default policy). Parameter *lRule* represents the locality policy; its value is either *min*(), *max*(), or *identity*(), which represent "the most specific authorization takes precedence, " "the most general authorization takes precedence, " or "no locality policy" modes, respectively. Parameter *mRule* takes three values *before*, *after*, or *skip*, which determines whether the majority policy is applied before the locality policy, after it, or not at all, respectively. Finally, parameter *pRule* represents the preference policy and determines whether positive or negative authorization is preferred in the case of remaining conflicts. (We assume that the subject hierarchy (SDAG) and the explicit access control matrix (EACM) are globally defined in the algorithm.)

In Line 1, relation *allRights* is created by calling Function *Propagate().* The details of this function are explained in the next section, but the effect is to apply the first three steps of the procedure described in the introduction to Section 3. An instance of relation *allRights* is illustrated in Table 1, where all the rights of subject *User* on object *obj* with respect to authorization *read* are depicted.

In Line 2, we check whether the caller is interested in applying the default authorization policy (*dRule* = "+" or "-") or not (*dRule* = "0"). In the latter case, only those rows of relation *allRights* are considered in which *mode* <> "d". In the former case (Line 3), those rows of relation *allRights* in which *mode*="d" are updated with the value of *dRule* ("+" if positive authorization is to be the default policy, "-" otherwise).

In Line 4, if the majority policy should be applied before the locality policy, we count the number of positive authorizations and negative authorizations that exist in relation *allRights*; however, as stated in Line 5, if the majority policy should be applied after the locality policy, we first apply the locality on relation *allRights*, and then count the number of positive and negative authorizations. In either of these cases (Line 6), the algorithm returns the authorization which is in majority.

```
Algorithm Resolve(s, o, r, dRule, lRule, mRule, pRule)
¤ To check whether subject s has a positive or a negative authorization for right r on object o
¤ Default rule dRule ∈ {"+", "-", "0"}
¤ Locality rule lRule ∈ {max(), min(), identity()}
¤ Majority rule mRule ∈ {"before", "after", "skip"}
¤ Preference rule pRule ∈ {"+", "-"}
¤ SDAG (subject hierarchy) and EACM (explicit access control matrix) are globally defined.

Output: either "+" or "-"

1. $allRights \leftarrow Propagate(s, o, r, SDAG, EACM)$;
2. if $dRule$ = "0" $allRights \leftarrow \sigma_{mode<>\text{"d"}}\, allRights$
3. else update $allRights$ set $mode = dRule$ where $mode$="d";
4. if $mRule$ = "before" {$c_1 \leftarrow \Pi_{count()}\left(\sigma_{mode=\text{"+"}}\, allRights\right)$; $c_2 \leftarrow \Pi_{count()}\left(\sigma_{mode=\text{"-"}}\, allRights\right)$}
5. if $mRule$ = "after" {$c_1 \leftarrow \Pi_{count()}\left(\sigma_{mode=\text{"+"},\, dis=lRule(dis)}\, allRights\right)$; $c_2 \leftarrow \Pi_{count()}\left(\sigma_{mode=\text{"-"},\, dis=lRule(dis)}\, allRights\right)$}
6. if $mRule$ <> "skip" { if $c_1 > c_2$ return "+"; if $c_2 > c_1$ return "-"; }
7. $Auth \leftarrow \Pi_{mode}\left(\sigma_{dis=lRule(dis)}\, allRights\right)$;
8. if $count(Auth) = 1$ return $Auth$;
9. return $pRule$;
```

Fig. 4. Algorithm *Resolve ()*

If neither positive nor negative labels is in the majority or the majority policy is not designated at all, we apply the locality policy to relation *allRights* to select its relevant rows (Line 7); if *lRule = min()*, only rows in which the value of column *dis* is equal to the minimum distance (the most specific authorizations) are selected; similarly, if *lRule = max()*, only rows in which the value of column *dis* is equal to the maximum distance (the most general authorizations) are selected; however, if *lRule=identity()*, all rows are selected (this is equivalent to no locality policy being designated).

Next, the values of column *mode* of corresponding rows are projected to form a set called *Auth*, which may be empty or contain positive and negative authorizations. If only one type of authorization is present (Line 8), it is returned; otherwise, the preferred authorization (*pRule*) is returned and the algorithm ends.

Table 3 illustrates the result of Algorithm *Resolve()* applied to our motivating example for several illustrative strategies. In particular, we trace the algorithm for eight strategy instances (selected from Table 2) namely D^+LMP^+, D^-GMP^-, D^-MP^-, D^-LP^+, D^+GP^-, P^-, GMP^-, and MGP^-. Table 3 shows values of c_1, c_2, *Auth*, and the effective mode derived by the algorithm, as well as its corresponding return line number. In the table, *n/a* means that the algorithm does not use the corresponding variable for the conflict resolution.

For instance, if one chooses the strategy instance D^-GMP^-, all default values of relation *allRights* are replaced with "-" (Line 3). Since the global mode of the locality policy is in place and there are one positive and one negative authorization from distance 3 in Table 1, both c_1 and c_2's values are assigned the value 1 (Lines 5). Then, since neither positive nor negative is in majority, the algorithm continues to Line 7, and *Auth* is assigned the value {+,-}. Finally, since there is a conflict in *Auth*, Line 9 of the algorithm returns the value of preference policy, which is "-" (indicated by P^- in the strategy instance), as the final derived decision with respect to triple <*User*, *obj*, *read*>.

As another example, if one chooses strategy instance MGP^-, in Line 2, only those rows of relation *allRights* in which the mode is not "d" are selected. Then, since the globalization policy is in place and there is one explicit positive authorization from distance 3 in relation *allRights*, the value of c_1 is set to 1 and the value of c_2 is set to 0 in Lines 4. Finally, the algorithm returns "+" from Line 6 as the final derived decision with respect to triple <*User*, *obj*, *read*>.

Table 3. Trace of *Resolve()*

Strategy	c_1	c_2	*Auth*	*mode*	Line
$D^+L\mathbf{M}P^+$	2	1	*n/a*	+	6
$D^-GM\mathbf{P}^-$	1	1	+,-	-	9
$D^-\mathbf{M}P^-$	2	4	*n/a*	-	6
$D^-L\mathbf{P}^+$	*n/a*	*n/a*	-,+	+	9
$D^+\mathbf{G}P^-$	*n/a*	*n/a*	+	+	8
$\mathbf{G}MP^-$	1	0	*n/a*	+	6
$\mathbf{P}^-$	*n/a*	*n/a*	-,+	-	9
$\mathbf{M}GP^-$	1	0	*n/a*	+	6

3.2 Function *Propagate*()

In this section, we explain the details of Function *Propagate()*, which returns all corresponding authorizations of a given subject, object, and authorization, shown as <*s*, *o*, *r*> when called from Line 1 in Algorithm *Resolve()*. The idea is to extract the part of the subject hierarchy in which *s* is the only sink. Then, using top-down breadth-first propagation, all authorizations from root subjects are propagated towards *s*. If a root subject has no authorization assigned in the explicit matrix, a letter "d" is assigned to it to represent the default authorization. Moreover, the distance of each authorization to *s* is computed, so that it can be exploited by Algorithm *Resolve()* if the locality policy is applied. Note that authorizations are propagated from *all paths* starting from the source node and ending at the destination. For instance, in Figure 3, authorization + from subject S_2 is propagated to subject *User* along two paths: $S_2 \rightarrow User$, and $S_2 \rightarrow S_3 \rightarrow S_5 \rightarrow User$.

Figure 5 illustrates Function *Propagate()*, which inputs parameters *s*, *o*, and *r* (representing the subject, object, and authorization on which the conflict should be resolved), as well as *SDAG* and *EACM*, which represent the subject hierarchy and the explicit access control matrix, respectively.

```
Function Propagate (s, o, r, SDAG, EACM);
¤ To obtain all authorizations, with respect to triple <s, o, r> by propagating explicit authorizations in EACM through subject hierarchy SDAG
¤ EACM has attributes <subject, object, right, mode>
¤ SDAG has attributes <subject, child>
¤ $ancestors(s) = \{s\} \cup \{x | \exists y <y,s> \in SDAG \wedge x \in ancestors(y)\}$
Output: table allRights
1. $SDAG' \leftarrow \sigma_{subject \in ancestors(s), child \in ancestors(s)} SDAG$;
2. $i = 0$;
3. $P \leftarrow \Pi_{subject, object, permission, i, mode} (SDAG' \bowtie \sigma_{permission=r, object=o} EACM)$;
4. $Roots \leftarrow \Pi_{subject} SDAG' - \Pi_{child} SDAG' - \Pi_{subject} P$;
5. $P \leftarrow P \cup Roots \times \{<o, r, i, \text{"d"}>\}$;
6. $P' \leftarrow \sigma_{subject \neq s} P$;
   repeat
7.     $i = i + 1$;
8.     $P' \leftarrow \Pi_{child, object, permission, i, mode} P' \bowtie SDAG'$;
9.     $P \leftarrow P \cup P'$;
10.    $P' \leftarrow \sigma_{subject \neq s} P'$;
11.    until $P' = \emptyset$ ;
12. return $\sigma_{subject=s} P$;
```

Fig. 5. Function *Propagate()*

In Line 1, we extract from *SDAG* the maximal connected sub-hierarchy *SDAG'*, in which *s* is the sole sink. In our motivating example, this line extracts the hierarchy depicted in Figure 3 from the hierarchy depicted in Figure 1. We set the initial depth of the iteration to 0.

In Line 3, we create a relation *P*, which consists of five columns namely, *subject*, *object*, *right*, *dis*, and *mode*. Values for columns *subject*, *object*, *right*, and *mode* are taken from the corresponding ones in relation *EACM* (as explained in Section 1). Column *dis* represents the distance of the explicit authorization from the subject, and takes its value from the iteration number *i*. Thus, the *dis* value for explicit authorizations is 0, for an authorization inherited from a parent it is 1, and for an authorization inherited from a grandparent it is 2. The first two lines of Table 4 show the result of Line 3 on our motivating example. Before completing the Function *Propagate()*, relation *P* will record all relevant authorizations propagated from all subjects to all other nodes.

In Line 4, we store all unlabelled root subjects of *SDAG'* into a relation called *Roots*. For instance, *Roots* contains $\{S_1, S_6\}$ if applied to our motivating example. In

Line 5, for each root subject with no explicit authorization with respect to *o* and *r*, we insert an additional row into relation *P* to assign it the default authorization. This results in the third and fourth entries in Table 4. In Line 6, we select as *P'* all identified authorizations other than those on the sink node *s*.

In Lines 7 to 11, we iteratively propagate all of the newly identified authorizations to all of the children of the corresponding nodes, stopping when no more nodes exist in *P'*. This involves increasing the distance by 1 (Line 7), copying the authorizations from each node to its children (with the increased distance) (Line 8), inserting the new authorizations into P (Line 9), and re-determining which authorizations still need to be propagated further (Line 10). The remainder of Table 4 shows the result of these lines on our motivating example. Finally, Line 12 selects and returns authorizations that correspond to subject *s*, which are shown in Table 1 for our example.

Table 4. All *read* authorizations on *obj*

subject	object	right	dis	mode
S_2	obj	read	0	+
S_5	obj	read	0	-
S_1	obj	read	0	d
S_6	obj	read	0	d
User	obj	read	1	+
S_3	obj	read	1	+
User	obj	read	1	-
S_3	obj	read	1	d
User	obj	read	1	d
S_5	obj	read	1	d
S_5	obj	read	2	+
S_5	obj	read	2	d
User	obj	read	2	d
User	obj	read	3	+
User	obj	read	3	d

3.3 Computational Analysis

The performance of Algorithm *Resolve()* depends on the structure of the subject hierarchy, on the placement of the explicit authorizations in the explicit access control matrix, and on the choice of subject, object, and right. We will examine the performance in practice in the next section, but here we summarize its asymptotic behavior in the worst case.

Consider first the structure of the subject hierarchy as represented by *SDAG*. Let ***r*** be the number of roots of the graph and let ***n*** be the total number of subjects in the hierarchy. We assume that at most one authorization is explicitly given for every subject-object-right triple; duplicates are meaningless and contradicting authorizations can be assumed to be disallowed. Thus, when selected subjects from *SDAG* are matched with explicit authorizations for a given object and right (Line 3 of Function *Propagate()*), at most one explicit authorization is joined to each subject in the subject hierarchy. Let ***p*** be the number of subjects assigned explicit authorizations for the given object-right pair. Finally, let $\boldsymbol{d}$ be the sum of the path lengths for *all* paths leading from a root or an explicitly authorized subject to the given subject of interest s.

Algorithm *Resolve()* first calls Function *Propagate()*. Lines 1 through 6 take time $O(\boldsymbol{n})$ to select a subset of the subjects, attach the explicit authorizations, and set the defaults in the remaining roots. Each authorization (default or explicit) is then pushed down each path to the node representing the given subject. The loop from Lines 7 though 11 of Function *Propagate()* thus require $O(\boldsymbol{d})$ time in total. Finally relation P contains all these propagated authorizations, but only those associated with the given subject s are returned; this returned relation includes exactly one tuple for each explicit authorization and at most one tuple for each root. In summary, Function *Propagate()* takes time $O(\boldsymbol{n}+\boldsymbol{d})$ and returns a structure of size $O(\boldsymbol{p}+\boldsymbol{r})$.

The remainder of Algorithm *Resolve()* repeatedly examines subsets of the relation *allRights*, and thus each line requires time at most $O(\boldsymbol{p}+\boldsymbol{r})$. Thus the total time for executing Algorithm *Resolve()* is $O(\boldsymbol{p}+\boldsymbol{r}+\boldsymbol{n}+\boldsymbol{d})$. Clearly $\boldsymbol{p}+\boldsymbol{r}$ is $O(\boldsymbol{n})$, so the complexity can be stated more simply as $O(\boldsymbol{n}+\boldsymbol{d})$. Unfortunately, since the number of paths in a directed acyclic graph can grow exponentially in the number of nodes in the graph, $\boldsymbol{d}$ is $O(\boldsymbol{n}2^{\boldsymbol{n}})$ in the worst case. We shall see that in practice, however, the algorithm is typically much better behaved, as subject hierarchies seldom contain the repeated diamond patterns that cause the number of paths to explode.

4 Experiments

We tested our algorithm first on synthetic data. We constructed several random complete directed acyclic graphs. In particular, KDAG(n) includes n nodes, one of which is a root and one of which is a sink, and $\binom{n}{2}$ edges (an edge between every pair of nodes), directed in such a way as to prevent cycles. Thus such graphs contain many more paths than would be expected in typical applications, and constitute good stress tests for the algorithm.

We executed our algorithm on random KDAGs of three different sizes. For each graph, we assigned explicit authorizations to subjects at random, choosing subjects proportionally to the number of members. In particular, 0.5% to 10.0% of the graph's *edges* were selected at random and their source nodes were assigned explicit authorizations. We then measured the CPU time for computing the result of the Function *Propagate()* (that being the dominant part of the algorithm) for each of the resulting SDAG-EACM pairs, and averaged over 20 random repetitions with the same parameters. Our experiments show that for small authorization rates (which often occur in practical cases), the running time is linearly proportional to the authorization rates (see Figure 6).

We also evaluated our algorithm on the subject hierarchy extracted from an installation of Livelink, Open Text's enterprise content management system[1]. In Livelink, groups can be arbitrarily structured and nested to arbitrary depth. In the environment we tested, the subject hierarchy has over 8000 nodes and 22,000 edges. There are 1582 sinks (individual users), each of which represents a real-world sample for our experiments. The depths of the induced sub-graphs range from 1 to 11.

[1] http://www.opentext.com/

We measured the time of our algorithm for each of the sinks in the Livelink subject hierarchy, using an authorization rate of 0.7% of the edges as above. The results are presented in Figure 7(a), plotting the CPU time as a function of ***d***, the sum of the lengths of all paths from explicit and default authorizations to the selected sink.

Figure 7(a) also compares the execution time of the *Resolve()* algorithm to that of the *Dominance()* algorithm, presented in Chinaei and Zhang's work [2]. The latter algorithm was designed to evaluate the D^-LP^- strategy as efficiently as possible under the assumption that there are relatively few explicit authorizations (i.e., that the authorization rate is low). Thus the comparison sheds some light on the overhead imposed by adopting a unified conflict resolution algorithm. It is important to note that the *Dominance()* algorithm is dependent on the placement of negative authorizations whereas the *Resolve()* algorithm is not. To account for this, we calculated the average of three trials for each data point for the *Dominance()* algorithm: one where 1% of the explicit authorizations are negative, one where half of them are negative, and one where all explicit authorizations are negative.

The *Dominance()* algorithm is occasionally very fast due to visiting an early negative authorization in the hierarchy, but it is not as efficient as *Resolve()* for objects that have few negative authorizations. Figure 7(a) shows that the run time for the *Dominance()* algorithm can fall anywhere below the time for the *Resolve()* algorithm, and occasionally it can be higher. On average, over all graph sizes and shapes in these experiments, *Resolve()* required 1260 ms to compute whether or not a sink subject was authorized to access an object, whereas the *Dominance()* average is 920 ms. Thus the flexibility to compute the value for any strategy comes at a cost of 27% overhead.

Finally, Figure 7(b) restates the behavior of Algorithm *Resolve()* against the number of nodes in the sub-graph rather than the total length of all paths in the sub-graph. The results show that graphs with very many subjects do not necessarily require much more time to resolve than do small graphs. From this data we conclude that it is unlikely that the asymptotic worst case performance will be problematic in practice.

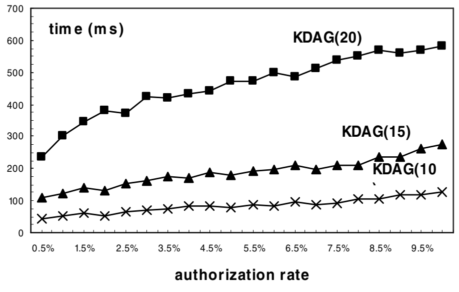


Fig. 6. Function *Propagate()* on synthetic data

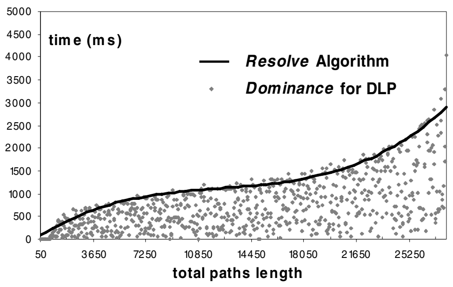


Fig. 7(a). Algorithms *Resolve()* and *Dominance()* on Livelink data

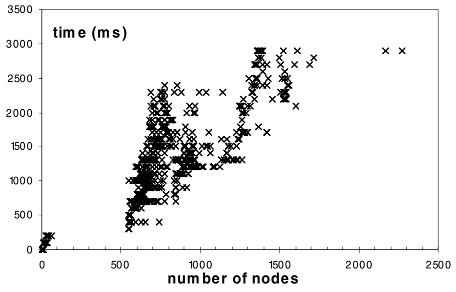


Fig. 7(b). Total paths lengths vs. number of nodes

5 Related Work

Bertino et al. propose an authorization mechanism for relational models in which conflicts are mainly resolved based on "the negative authorization takes precedence" policy [1]. They also introduce the concept of weak and strong authorizations, which is equivalent to using our combined strategy instance D^-LP^-.

Jajodia et al. use Datalog programs to model access controls of hybrid authorizations with a wide range of conflict prevention/resolving policies [7]. Their modeling stores the raw authorizations and computes the effective authorizations for a <*subject, object*> pair in time linear to the size of the Datalog program (rules and ground facts). However, their ground facts include the transitive closure of the subject hierarchy (which cannot be computed in linear time) plus all the raw authorizations. The potentially large number of ground facts implies that even a linear time solution may not be efficient in practice. To answer access control queries efficiently, they suggest materializing the entire effective access control. The accessibility check for a given <*subject, object*> pair is thus equivalent to checking the materialized effective access control table (constant time). However, considering the formidable size of the effective access controls, which is the product of the number of objects and the number of subjects, this approach is not practical for very large systems. Moreover, the materialized effective access controls are not self-maintainable with respect to updating the explicit authorizations, and even a slight update to the explicit authorizations could trigger a drastic modification to the effective ones, making the maintenance task very expensive.

Some existing solutions for computing effective authorizations assume that the explicit authorizations are propagated on tree-structured data [4, 12, 14, 15]. This trivializes conflict resolution since there is only one path between any ancestor and a leaf. Moreover, the number of ancestors for a leaf is bounded by the depth of the tree, which is usually a small value in real world data [11]. Unfortunately, real world subject hierarchies are mostly DAG-structured rather than trees: the UNIX file system allows a user to be member of several groups at the same time, and in role-based access control systems, a user can be assigned several roles and each role can be assigned to multiple parent roles [5]. When explicit authorizations are propagated on a

DAG subject hierarchy, a sink subject potentially has all subjects as its ancestors, and each ancestor may have several paths reaching to that sink. Therefore, none of the approaches for tree-structured data are appropriate in this setting.

Cuppens et al. propose a conflict resolution model for documents containing sensitive information [3]. They address the problem of downgrading the classification of these documents when their contents become obsolete. Their approach is to impose a strict order of preference between rules and does not include any hierarchy among subjects.

Koch et al. provide a systematic graph-based conflict detection and resolution algorithm based on two properties namely, *rule reduction* and *rule expansion* [9]. Using these properties, they transform a conflicting graph into a conflict-free one. However, their approach is applicable only to the rules that are related to one another, whereas our approach addresses independent policies.

Finally, our approach is also different from the combining algorithms in XACML [12], in which the resolution model relies on the data hierarchy rather than the subject hierarchy.

6 Conclusions

In this work, we have extended Chinaei and Zhang's conflict resolution framework for hybrid authorizations, by investigating legitimate combinations of a variety of popular conflict resolution policies. Our framework includes a variety of resolution models to support both closed and open systems, as well as both restricted and relaxed applications. Most importantly, we have enhanced this framework with a unified parameterized algorithm in which a wide range of combined strategy instances are simultaneously supported. This framework inspires access control system providers to separate the conflict resolution component from the system itself. With such an approach, system buyers no longer need to replace the whole system when they decide to change the access control policy. Instead, security administrators can trigger a chosen strategy, among many, without needing to reinstall the whole system. Moreover, access control system providers can sell the same system to all users, regardless of their specific access control needs.

We have experimented with different sets of data layouts. Our pilot experiments show that our algorithm incurs little overhead and performs well when the explicit authorization rate is below 10%, which is appropriate for most practical applications.

We propose several directions to continue this work. First, since there are many nodes in the subject hierarchy that are ancestors of several sinks, it would significantly improve the performance of the algorithm if the derived authorizations of such nodes stored in a cache for later uses. Second, in this work we exploit the subject hierarchy only. It is important to support mixed hierarchy of subjects and objects. Third, our framework can be augmented to support three different propagation modes: when propagating an authorization along a path and meeting another authorization on that path, either the first, the second, or both authorizations could be propagated further. Fourth, we suggest to enhance the framework by adding other access control constraints such as separation of duties and conflict of interests [8, 13]. Lastly, it is interesting to optimize the *Resolve()* algorithm for special purposes of covering a smaller subset of combined strategies for applications in which performance is more important than complete flexibility.

Acknowledgments

We gratefully acknowledge the Natural Sciences and Engineering Research Council of Canada, Communications and Information Technology Ontario, Open Text Corporation, the Mathematics of Information Technology and Complex Systems, and the University of Waterloo for their financial support.

References

1. Bertino, E., Jajodia, S., Samarati, P.: A flexible authorization for relational data management systems. ACM Transactions on Information Systems 17(2), 101–140 (1999)
2. Chinaei, A.H., Zhang, H.: Hybrid authorizations and conflict resolution. In: Jonker, W., Petković, M. (eds.) SDM 2006. LNCS, vol. 4165, pp. 131–145. Springer, Heidelberg (2006)
3. Cuppens, F., Cholvy, L., Saurel, C., Carrere, J.: Merging Security Policies: Analysis of a Practical Example. In: Proceedings of the 11th Computer Security Foundations Workshop, pp. 123–136 (1998)
4. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for XML documents. ACM Transaction on Information and System Security 5(2), 169–202 (2002)
5. Ferraiolo, D.F., Kuhn, D.R.: Role Based Access Control. In: Proceeding of the 15th NIST-NCST National Computer Security Conference, pp. 554–563 (1992)
6. Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in Operating Systems. Communications of ACM 19(8), 461–471 (1976)
7. Jajodia, S., Samarati, P., Sapino, M.L., Subrahmanian, V.S.: Flexible Support for Multiple Access Control Policies. ACM Transactions on Database Systems 26(2), 214–260 (2001)
8. Joshi, J., Bertino, E., Sahfiq, B., Ghafoor, A.: Dependencies and Separation of Duty Constraints in GTRBAC. In: Proceeding of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT '03), pp. 51–64. ACM Press, New York (2003)
9. Koch, M., Mancini, L.V., Parisi-Presicce, F.: Conflict Detection and Resolution in Access Control Specifications. In: Proceedings of the 5th International Conference on Foundations of Software Science and Computation Structures, pp. 223–237 (2002)
10. Lampson, B.W.: Protection. In: Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems, pp. 437–443 (March 1971)
11. Mignet, L., Barbosa, D., Veltri, P.: The XML Web: A First Study. In: Proceedings of the International World Wide Web Conference, pp. 500–510 (2003)
12. Moses, T.: eXtensible Access Control Markup Language Version 2.0, Technical Report, OASIS (February 2005)
13. Nyanchama, M., Osborn, S.L.: The Role Graph Model and Conflict of Interest. ACM Transaction on Information Systems Security 2(1), 3–33 (1999)
14. Yu, T., Srivastava, D., Lakshmanan, L.V.S., Jagadish, H.V.: Compressed Accessibility Map: Efficient Access Control for XML. In: Bressan, S., Chaudhri, A.B., Lee, M.L., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, pp. 478–489. Springer, Heidelberg (2003)
15. Zhang, H., Zhang, N., Salem, K., Zhuo, D.: Compact Access Control Labeling for Efficient Secure XML Query Evaluation. In: Proceedings of the 2nd International Workshop on XML Schema and Data Management (2005)

Multi-layer Audit of Access Rights

Birgit Pfitzmann

IBM Zurich Research Lab,
Säumerstr. 4, CH-8803 Rüschlikon, Switzerland
bpf@zurich.ibm.com

Abstract. In the context of regulatory compliance, the question is often whether an enterprise can guarantee that only certain people can access certain data or perform certain business functions on them. Examples are controls over financial data in Sarbanes-Oxley and access to personal information in privacy laws such as HIPAA and the California Senate Bill 1386. Such guarantees also have to be strictly audited. For individual access control systems, such questions are standard at least in theory. However, to the best of our knowledge such questions have never been addressed for entire system stacks containing multiple layers of data representation with potentially different access mechanisms. For instance, financial data may be accessed by using an access right to the official financial application, but also by using an administrator right to an underlying database or by logically or physically accessing an unencrypted backup tape with the data. We propose an overall model and algorithms to deal with this situation. We study both advance queries for validating a proposed system and a posteriori queries in audit, problem determination, or litigation.

1 Introduction

In the last few years, new regulations have had a strong impact on security management in enterprises and posed a number of new challenges in security implementation, evaluation, and audit. One reason for this profound impact is the increased punishment for misbehavior, such as personal responsibility of CEOs in the Sarbanes-Oxley Act and the obligation to personally inform customers when an enterprise has lost their personal data in the California Senate Bill 1386 and subsequent similar regulations in many other US states. A second reason for the strong impact is that many new laws come much closer to making technical requirements than older laws. This does not mean that they prescribe a specific technology such as a specific access control mechanism or cryptographic algorithm. However, they prescribe that enterprises document the methods they use, demonstrate the effectiveness of these methods, and support detailed auditing. Fulfilling all these requirements for information stored digitally and for processes carried out with machine assistance is almost impossible without strict IT security measures.

A specific need that often arises when the higher-level regulatory requirements are broken down to the IT level is to evaluate who can access what information.

For instance, for Sarbanes-Oxley compliance one needs to know who has write access to financial data. For privacy laws, one needs to know who has read access to personal information.

At a first glance, this question may seem trivial to answer: Most enterprises have access control systems in place for most data repositories and applications. Hence to answer questions about who has access to certain data, one “simply” looks up the access control policy and analyzes the rights to the information under consideration. We put “simply” in quotes because with existing tooling even this is not trivial in many cases. However, we will look one step further: Usually, the “official” access control system to a certain type of business information, such as access control integrated into a workflow or access control to the database records, is by far not the only path to the information. For instance, instead of using an official financial application, one may also access financial data by using an administrator right to an underlying database. Or one might be able to access an unencrypted backup tape with the data, either logically or by physically stealing it. To the best of our knowledge, there are no models or systems in theory or practice for evaluating such overall access control situations. We provide such models and foundational algorithms on them in this paper.

We call this *multi-layer access control* or, when auditing existing access rights is the main issue, *multi-layer audit of access rights.* We introduce a static model for the situation, consisting of a base model and a number of extensions that are needed for certain more complex applications. We also extend this to a dynamic model, i.e., to cover the evolution of a system over time. Based on these models, we propose algorithms to answer the typical audit questions. Again, we start with the static case, where the question is about current access rights, or rather access rights at any particular point in time for which a static multi-layer access control model is given. After that, we investigate queries about the past and the future. Queries about the past come in two flavors: One can ask either who actually *has accessed* data, or who *could have accessed* them. Both questions are important for auditing purposes, because one may have to report and remediate a lack in oversight and governance even if nothing bad happened because of it. Queries about the future are about who could access certain data in the future, given the knowledge about potential changes in the system and the rights. This is a multi-layer extension of the well-known safety questions for access rights. We believe this extension will give rise to significant new work in this space beyond the start that we take here.

Beyond typical all-or-nothing access control models, our model takes encryption into account, because in real enterprises, data encryption is currently becoming a standard measure to address some of the new regulatory concerns about data access. Initially, encryption is primarily used for data on tapes and for data in transit on the Internet, but the use will certainly extend to more types of data representations in the future. The model extensions we make to cover encryption also cover other shared protection mechanisms such as 4-eye principles in the access control system and secret sharing.

Furthermore, our model can not only address integrity and privacy, but also availability. Availability of certain types of data is also becoming strongly regulated at present under the name of *data retention*.

Overview of this paper. In Section 2, we review underlying concepts of access control, and we discuss some concepts whose names sound similar to our multi-layer access control, but which are something else. In Section 3, we present the new model that allows us to treat multi-layer access control and audit. We present a basic static model, extensions to the static model, and models for the dynamic development of the model over time. In Section 4, we present algorithms for evaluating access rights over the static model. Section 5 contains algorithms for evaluating questions about past access rights, first for actual accesses and then for potential accesses. Section 6 contains algorithms for evaluating questions about potential future access rights. Section 7 contains proposals for instantiating a multi-layer access control model for a given real-world enterprise. We finish with a conclusion in Section 8.

2 Related Work

We start with a short survey of basic work on static and dynamic access control considerations. We proceed with the closest related work, that on access control in an interplay of applications or workflows with databases. After that, we mention some existing concepts whose name sounds similar to our multi-layer access control and audit, e.g., multi-policy systems and multi-level security. However, these concepts are actually quite different.

Static access control. Access control is one of the oldest topics in security research. An access control scheme is typically seen as a matrix, often 3-dimensional with dimensions called subjects, objects, and actions, and with Boolean values describing whether access is allowed or not. Such a matrix is a *static* access control model, describing the access rights at one point in time. An early paper in this space is [10], but the concepts probably existed in specific operating systems before. Typically, a static access control *language*, such as the UNIX access control rights or the recent XACML standard, does not write out an access control matrix field by field, but contains abbreviations such as user groups or user roles and object hierarchies. There are also other extensions, e.g., 3-valued logic with “don’t-care” values, additional dimensions like “purpose” in privacy policies, condition expressions, and post-access obligations. While we are unable to say for each of these concepts where it was first proposed, some influential early papers with them are [19,21,23,9,16,11,15]. A book summarizing many access control models is [7].

Dynamic access control. The seminal paper for dynamic access control theory is [12]. It introduced the notion of access control commands, small procedures built from basic access control matrix changes that can only be executed as a whole. Hence one can consider the possible evolution of access rights. This paper also introduced the safety problem, which is precisely a query whether a certain subject may be able to access a certain object in the future. The undecidability of this problem for general protection systems spawned a large body of research on restricted systems, i.e., restricted command sets, e.g., [14,22,17].

Access control with databases and applications. Access control with two to three layers has been considered specifically for the interplay of applications and/or workflows with underlying databases, e.g., in [20] or industrially in [6]. However, to the best of our knowledge the questions that have been treated in this context are quite specific to these two layers, e.g., to what extent database access control enforcement can be used to mirror dynamic access constraints defined for a workflow, or how one can find out from a given application what access rights its individual users needs.

Multi-policy systems. These are systems where one access control enforcement point, at one level of abstraction, takes several policies into account [13]. Topics in this context are whether the policies are combined, typically with the Boolean AND operator, into one policy or whether the access control enforcement point queries multiple access decision engines. The policies may all be in the same policy language or in different languages. In the former case, policy algebras are of particular interest, such as in [5,1].

Federated access control. This means that access control systems under different domains of control are considered [8]. However, these domains are at the same level of abstraction, while we consider levels of abstraction. Typical examples are digital rights management (DRM) and privacy. For instance, in DRM one may consider an audio file flowing from a producer over a distributor to a media player. These are three domains of control, and one desires that the distributor and the media player honor the policy of the producer. Similarly, in privacy one desires that personal data are only forwarded with the original usage restrictions that had been promised to the concerned person by the initial data collector. Both DRM and privacy are of course also considered outside the name of federated access control.

Policy hierarchies. This term from [18] refers to policy refinement, originally for a broader scope of management action policies, of which access control policies are a very specific case. Refinement specifically for access control has also been considered in detail in the privacy context in [2].

Multi-level security. This denotes a specific class of access control policies, first formalized in [3], originating from military applications. The levels have nothing to do with our layers, but are security levels such as “highly confidential”, “confidential”, and “public”.

Object hierarchies in access control. Finally, there is the standard notion of object hierarchies in static access control policies. This refers to subject or object hierarchies at one level of abstraction. For instance, an object hierarchy might first distinguish patient data into medical versus financial data, and then further classify medical data into normal medical data versus mental health data, or data of children versus data of adults. One does not suddenly consider different abstraction layers such as tapes in such a classification. In particular, access control algorithms dealing with object hierarchies are designed for being handled by one access control system, while we deal with a situation where there are different access control systems at different abstraction layers.

3 Model of Multi-layer Access Control

The main novelty of our model is that we consider a data item not only in one incarnation (typically the highest available abstraction, such as an application object or a database entry), protected by one access control system, but in its different incarnations on the different system layers underlying this abstraction. We use the term *incarnation* to cover both a physical version of a data item (e.g., two backup copies are two incarnations) and, more importantly, a logical version on a specific layer of abstraction. For instance, three incarnations of a database record may be the logical view of it as a database record, the file containing the database in the operating system, and the blocks on the storage system that hold this file. This multi-layer view is crucial for real-life access control and audit because the incarnations are typically managed by different access control systems, and many real-life security incidents occur because of insufficient control on the lower layers. Examples are the widely publicized incidents where enterprises lost tapes with unencrypted personal data, and the current problems to restrict the administrator rights on systems that hold personal data. The layers of abstraction that define the incarnations are the typical abstraction layers of current hardware and software stacks.

In this section, we first present a basic static model for multi-layer access control, i.e., we first concentrate on one point in time. Then we present extensions to this static model. Afterwards, we introduce dynamic versions of the model as needed for later considering queries about the past and the future.

3.1 Basic Static Model: Data-and-Policy Tree

Our core model structure is a graph of incarnations of a piece of data under consideration. In the simplest case, this graph is a tree:

– The tree root is the top-level data item D_{top} about which one might set an "original" access control policy or get an audit query. For instance, D_{top} may be a financial statement or a set of personal data about one person.
– The children of a node D represent all the incarnations of D on the next lower abstraction layer.

We call this a *data tree*. In the static model that we define in this subsection, the data tree refers to one specific point in time. In principle, D_{top} can be on an arbitrary layer of abstraction. In the motivating examples of regulatory compliance it is typically a business-level data item.

A *data-and-systems tree* extends each node D of a data tree by the name S of the system that includes the access control to this data incarnation D. Thus the nodes of a data-and-systems tree are pairs (D, S). Only the root D_{top} of a data tree may be without such a system, because the business-level policy or the audit query may be made at a level of abstraction which is not directly represented by an actual system. At least for the leaves of a complete data-and-systems tree, the systems will be physical and not digital.

A *data-and-policy tree* extends each node (D, S) of a data-and-systems tree by an access control policy P, representing the access control policy of S at the considered point in time. Thus a data-and-policy tree has nodes of the form (D, S, P).[1] The policies may be in different policy languages; here we consider the policy language indication as part of the policy. If S is a physical system, its access control policy will often not be formalized a priori, but simply realized by fences, door locks etc. that are only documented informally. However, in principle these are also access control mechanisms that can be formalized by individual static access control policies as described in Section 2.

Example. Let our top-level data item D_{top} be a check in the financial sense. Figure 1 shows the data-and-systems tree for this example. In a bank, the check might be handled in two applications, as a bitmap image in an imaging application and as an XML document in a payment business process application. We call the image D_i and the XML document D_x, and the corresponding applications *Imaging* and *PayProc*. This gives us the children of the top-level check abstraction D_{top}. One layer below, the image D_i may be handled in a scanner *Scanner* and in a long-term database *ImgDB*. We call these incarnations of the image $D_{i,1}$ and $D_{i,2}$. We assume that the image database is stored on a disk and on two backup tapes. Hence there are three incarnations of $D_{i,2}$. We call them *Blocks*, $Bits_1$, and $Bits_2$ and the corresponding systems *DiskSys* and *TapeSys*, assuming that the same tape controller system controls both tapes. Similarly, we might consider the physical memory of the scanner and multiple incarnations of the XML document D_x; this is indicated by the symbols "..." in Figure 1.

3.2 Extensions of the Static Model

There are many situations in real life for which one has to extend the basic static model from Section 3.1, or at least use it in non-standard ways.

Child incarnations with coarser granularity. It is sometimes not easy to precisely identify the children of a data incarnation D. For instance, when we consider how the image $D_{i,2}$ in our check example is stored on the disk, i.e., what the incarnation *Blocks* really is, it consist of multiple blocks spread around on the disk or even on several disks. In such cases, it is sometimes useful not to try to find precisely the incarnation of D, but instead to consider the next coarser granularity of data that can be identified and protected individually on this

[1] For later query evaluation algorithms, D and P alone would suffice. In practice, the step via S is important to find P.

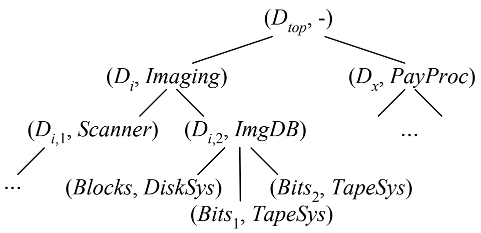


Fig. 1. Example of a data-and-systems tree

layer. In the example, this might be an entire disk or disk array instead of only the blocks corresponding to $D_{i,2}$.

Multiple systems for one data incarnation. Sometimes multiple access control systems exist for the same data incarnation. For instance, tapes on a truck on their way to a backup location may be in a safe, and the truck itself may be locked. At least in this example, we have a logical AND of the access control policies of the two systems, i.e., one needs to get into the truck and into the safe. We can consider the AND of these policies as one policy P, or allow a multi-policy representation per node in the data-and-policy tree.

Vulnerabilities. Many access control enforcement mechanisms can be broken in some way. For instance, an operating system may have vulnerabilities that give an attacker root rights, a physical lock may be physically broken, and logical systems may be circumvented by social engineering. We can consider this as conditional access where the breaking is a condition, possibly associated with the resources needed to perform it. The condition can be considered part of the policy P for the considered system in the data-and-policy tree, but it may be clearer to represent vulnerabilities as an OR of the official policy P and the breaking condition.

Overall user model. It is useful if the subjects of the different access control policies in a data-and-policy tree all come with mappings to an overall set *Users* of users. By "users" we mean real people, because responsibility ultimately lies with people. It is unlikely in real enterprises that the subject sets of the policies on different abstraction layers are all originally such a set *Users*, but the desired mapping might be given by an identity management system or a metadirectory. For the data-and-policy tree, one may replace the original subjects in each policy P by their real-user equivalents, but in practice one needs to consider the original policy and the mapping (identity management system) separately.

Non-user subjects. In some systems, the access control subjects may not be any encoding of users from the real user set *Users*, but applications, transactions, partner computers etc. For these systems, the mapping of policy subjects to subsets of *Users* will need a further hierarchical algorithm over additional information about these applications, transactions, or computers. For instance, one needs to identify the system S' that contains the access control policy P' to the application A that can access our data incarnation D under consideration.

k-of-n child relationships. Sometimes certain children $D_1, \ldots, D_n$ of a data incarnation D in the data tree do not each individually correspond to D, but only together. For this case, we define k-of-n child relations. The "n" means that there are n incarnations, and the "k" denotes how many of them are needed to provide what one would call "access to D". For instance, if D is stored in encrypted form, then the encrypted data D_1 and the key D_2 form a 2-of-2 structure of children if "access" means the ability to read. Normal child-relationships correspond to 1-of-1 structures.

The k-of-n situation may be extended to so-called general access structures, introduced in [4], where not all sets of data incarnations that provide access to a data item D are of the same size k.

Goal-dependent trees. The data trees may be different for confidentiality, integrity, and availability. In our encryption example, if D_1 and D_2 are the only incarnations of D on the next lower layer, then destroying one of them is enough to destroy D. Hence for availability, D_1 and D_2 constitute two normal 1-of-1 children of D, while we saw that they are a 2-of-2 system for confidentiality. (For integrity, they are also two 1-of-1 children unless the cryptosystem used also provides authenticity. In that case, the ciphertext D_1 is no child at all for integrity, only the key D_2). In particular, for auditing integrity and availability, the data tree can be pruned at data incarnations that are never written back or interpreted as the original, e.g., copies made to log files or data that are not erased on physical media, while for auditing confidentiality such copies must be considered.[2] Hence in general, we may have a *confidentiality data tree*, an *integrity data tree*, and an *availability data tree* for the same top-level data item D_{top}. If a data incarnation occurs in more than one of these trees, the corresponding system in the data-and-system trees will be the same. In the data-and-policy tree one can either use the same overall policies, or try to separate them according to actions with read, write, and destroy aspects.

3.3 Dynamic Model for Queries About the Past

So far we designed a model of data incarnations and the corresponding systems and policies for one point in time. This is sufficient for queries that refer to one point in time, such as "who can access the check D_{top} now?". However, there are also other queries. We first consider models for queries about the past, such as they often occur in audits. There are two ways of modeling the history, which will typically both be needed in practice:

[2] The log files, however, may be top-level data incarnations for integrity on their own. For instance, to answer the query "who could have changed this bank account statement" it is irrelevant who could have changed the log file, but the question "who could have changed the log of this bank account statement" is equally valid.

- Store data trees and actual accesses. This is the history type needed for queries like “who changed the check D_{top}”? The minimum data needed for this is a list of all accesses (D, s, a, T) where D is the directly accessed data incarnation, s the accessing subject, a the action performed, and T the data tree, or set of data trees, to which D belonged at the time of this access. Typically, additional data will be stored for each access to enable more query types, in particular the time t of each access, evidence of the access, arising obligations. Furthermore, one will usually the author and time of each policy change.
- Store data-and-policy trees. If one also expects questions “who *could* have accessed the check D_{top}?”, one needs to store all data-and-policy trees that occurred over time.

3.4 Dynamic Model for Queries About the Future

To evaluate whether a system is well-protected, it is not sufficient to consider the current access rights via its current data-and-policy tree. The reason is that most systems do not have one access control policy fixed forever, but certain ways of changing such a policy. These changes may be controlled by the system itself or on a different (typically lower) layer. For instance, changes to UNIX access control policies are protected by the same UNIX access control: File owners can change file access settings with the command “chmod”, and file ownership is a notion within the same access control model. Hence UNIX access control is essentially an instance of the theoretical treatment of dynamic access control as introduced in [12] and discussed in Section 2. In contrast, if access rights are just a file of deployment settings, e.g., in a firewall, they are administered separately by access to that file. In that case, the access control settings of each system S become an additional data object D with a data-and-policy graph.

Hence in the first case an individual policy P in the data-and-policy tree is already dynamic and does not need extra treatment. In the second case, an individual policy P is static and its changes are governed by another data-and-policy tree in which P is the top element D_{top}. Both cases may occur in the same data-and-policy tree.

Particular operations that must be considered in a multi-layer system beyond those of standard dynamic access control are the creation and deletion of data incarnations, i.e., changes to the data tree. The most important operation is the copy operation on a data incarnation D (possibly on a part of D only). In a confidentiality data tree, copying creates a sibling D' to D, and then a new sub-tree of lower abstractions of D' has to be created that reflects how D' is stored. Another important operation is moving of a data incarnation to another system (e.g., as in outsourcing or archiving).

We consider changes to a system S itself that may change the accessibility of data items (for instance, a database schema change) as special cases of access control policy changes.

Queries about the future like "who could ever manage to change the check D_{top}?" now have to be evaluated by, in principle, considering all potential evolutions of the set of data-and-policy trees.

4 Algorithms for the Static Case

As the main audit query, we consider a query "who can access D_{top}" for a possibly abstract data item D_{top}. Very roughly, the corresponding algorithms work as follows:

– Construct a data-and-policy tree T as defined in Section 3.1 with the root D_{top}.
– Answer the query for individual data incarnations D_i in T, yielding subject sets U_i.
– Combine the individual answers into an overall subject set U according to the structure of the tree T.

The second step is usually done for *each* data incarnation D_i in T. However, sometimes one may speed this up by interleaving with the third step.

If T is a normal data-and policy tree without k-of-n child relationships, the combination step is simply that the set U of subjects who can access D_{top} is the union of all the sets U_i. If I denotes the set of indices of the data incarnations in T, then in formulas $U := \bigcup_{i \in I} U_i$.

We now extend the treatment to trees with k-of-n child relationships in a bottom-up way.

For a k-of-n node D that has no other k-of-n nodes beneath it, one works as follows: Given the sets U_i for each child D_i (with $i = 1, ..., n$) of D, a single subject u can access D if $u \in U_i$ for at least k different values of i. A set U of subjects can access D together if there exist at least k different values of i for which at least one member of U is also a member of U_i. In formulas this means that there exists a subset $J \subseteq I$ with $|J| \geq k$ such that $U \cap U_j \neq \emptyset$ for all $j \in J$. Hence for such a node D we obtain a set $\mathcal{U}$ of authorized subsets U, i.e., of sets U that fulfill the formula we just gave. This is precisely what one calls a general access structure [4]. Above a k-of-n node in a tree, we therefore continue with general access structures, not only with individual users who can gain access.

For a normal node with k-of-n nodes underneath, the general access structure $\mathcal{U}$ is computed as follows: A set U can access D iff it can access at least one child incarnation D_i. Thus $\mathcal{U}$ is the union of the access structures $\mathcal{U}_i$ of the child data incarnations D_i.

For a k-of-n node with k-of-n nodes underneath, a set U can access D if and only if this set can access at least k child incarnations, i.e., if there exists $J \subseteq I$ with $|J| \geq k$ such that $U \in \mathcal{U}_j$ for all $j \in J$. Thus $\mathcal{U}$ is the set of the sets U fulfilling this formula. This finishes the investigation of all node types.

If we consider goal-dependent trees (see Section 3.2), this same algorithm may have to be applied to three different concrete data-and-policy trees for one top-level data item.

5 Algorithms for Queries About the Past

As explained in Section 3, we distinguish queries about actual past accesses and about potential accesses, i.e., about past access rights.

5.1 Queries About Actual Past Accesses

For queries about actual past accesses, a history of all accesses is necessary and sufficient.[3] This is the first model from Section 3.3.

For the query who accessed data item D_{top} one has to consider all data trees that occur in stored accesses and that contain D_{top}. If all these trees are normal (without k-of-n nodes), the set of users who accessed D_{top} is the union of the sets of users who accessed any node D below D_{top} in any of these trees.

For trees with k-of-n nodes, we first consider the case that there is only one such tree in the considered time period, i.e., the data item D_{top} is stored in the same way all throughout this time period. Then a user u or a user set U accessed a k-of-n node D iff they accessed k of its children within this time period. For the access to the children, one proceeds recursively in the same way.

Now we consider that are several trees for D_{top} in different sub-periods of the considered time period. For instance, additional copies might have been made on some abstraction layers in this time. Then if a certain k-of-n node D remains the same across several sub-periods, accesses from these two sub-periods must to be combined. For instance, a person who has read a ciphertext in one sub-period and the corresponding key in another sub-period has had read access to the cleartext data item. In other words, we have to proceed node by node with the algorithm described before, not sub-period by sub-period.

5.2 Queries About Potential Past Accesses

For a query about who *could* have accessed a data item in the past, a history of data-and-policy trees is sufficient. This is the second model from Section 3.3. Then one can in principle use static evaluation for every access right state that occurred in the time period for which the query is made. In practice, the efficiency of these evaluations can be improved upon.

6 Algorithms for Future Queries

As described in the introduction, future queries ask who may become able to access data D_{top}, including the changes to the systems and access control policies that might happen. This is an extension of the classical safety question of access control theory (see Section 2) to multi-layer systems.

[3] The only exception to this necessity is when it is clear that *someone* accessed the data and over the entire time, only one person could have done so, but this case seems rare.

If one keeps the question so simple, there will be many data in current enterprises where the answer is “everyone”. The reason is that many data are handled only with discretionary access controls, i.e., at least their current owner can give the data to everyone. For instance, the owner of a file in a normal file system can set world-wide access rights on the file, or send the file to anyone via email. Hence we refine the question in two steps:

– First, when asking “can a set U of users access data D_{top}”, one can consider only the system changes that the set U can make. This means considering only the immediate future as far as this user set U can influence things without help from anyone else.
– Secondly, if one looks further into the future, one should distinguish legitimate and illegitimate changes, in particular for the users *not* in the set U. This assumes that there are two different access control and system change policies (in particular, policies for the passing of data) in one or many systems: One policy P_{enf} that is actually enforced in the system, and another policy P_{hum} that has been communicated to the human users outside the system, e.g., a written guideline “only send medical data by email to other medical personnel”. The policy P_{hum} might even contain some multi-layer directives such as “only send medical data by email in encrypted form”. The refined query then becomes “can a set U of users access data D_{top} if all other users $u \notin U$ only make changes according to policy P_{hum}”. (The fact that each system enforces its policy part P_{enf} is implicit in all our considerations.)

Technically the query types where the users outside the considered set U do either arbitrary things or nothing can be regarded as special cases of the last query type with a policy P_{hum} that allows arbitrary actions or nothing, respectively.

Given such a query, in principle one has to follow the graph G of all possible developments of the data-and-policy tree for D_{top}. Once one has this graph G, one can answer the static query “who has access to D_{top} in T” for every tree T in G and form the union of the results (treating k-of-n nodes that occur in several trees as in the case of past queries). Constructing the graph G works as follows in principle: One initializes G by a graph with only one node, the original tree T_{orig}. Then one repeatedly considers each tree T in G that was not considered before (i.e., one marks each tree as considered or not). For this, one constructs every possible successor tree T' of T under the operations described in Section 3.4 and adds it to the graph G. Finding the successor trees of a tree T is done by considering every node of the tree and its possible successors. This includes duplicating or destroying the node or constructing new children of the node. The algorithm of determining successor systems will not necessarily terminate, and indeed it is known that even for single-layer systems with rather simple dynamic access control systems the problem can be undecidable. However, for audit purposes, if one can *not* show certain access restrictions, this indicates that access beyond these restrictions *could* be possible, and thus the system should typically be further restricted.

7 Constructing a Data-and-Policy Tree

A core new element of our model is data-and-policy trees, in particular their data-and-systems tree parts. For auditing in real enterprise applications, these trees must be derived from existing systems. This is not trivial, but certain existing enterprise models may help:

– If the enterprise has an explicit data model and if D_{top} occurs in this data model, then follow this data model for the incarnations of D_{top}. For instance, a good data model should state which business data are in which databases.
– If D_{top} is an abstract business item that does not occur in an explicit data model, then use the formal or informal business model to find out how this business item is instantiated into one or more actual data incarnations D, and then proceed as in the first bullet.
– Where the data model ends, follow the deployment model. This is the model about how applications are distributed onto middleware, and middleware onto systems and finally onto hardware. For instance, a good deployment model should state on which server a database resides, which disks are used, and how the backup is done. This is precisely the information needed for the lower levels of abstraction of a data-and-systems tree.

One should always check that the operating systems and the physical media have not been forgotten in a constructed data-and-systems tree.

If there is no data model or business model or deployment model, and if it is impossible from available documentation and knowledge to derive one, one should try to search for the data with mechanisms like full-text search. In this case, one may find incarnations at rather low levels, and may have to develop the higher layers of the data-and-systems tree upwards.

When considering the access control models of all the systems thus obtained, special roles such as “administrator” always have to be considered besides normal user roles, although in many systems these roles and their rights are predefined and do not show up in the explicit access-control policies.

8 Conclusion

We have introduced the concept of multi-layer access control and particularly of multi-layer audit of access rights. These concepts are key for addressing new security requirements imposed by legal regulations. Core concepts are data-and-policy trees containing data incarnations at different abstraction layers with different policies, and algorithms following those trees and, in the dynamic case, their evolution. We put special emphasis on allowing k-of-n nodes in the data trees in order to cover solutions employing encryption and other shared-control mechanisms whose use (although not very systematic yet) has strongly increased because of certain new regulatory requirements.

We have concentrated on auditing existing access control policies. An alternative approach is to set the policies of all data incarnations in one data tree according to a desired business-level policy for the top-level data element. Our model (Section 3) is also needed for this approach, and so are the methods to construct model instantiations (Section 7). In principle, such an approach is the end goal of enterprise-wide access control products. However, while such products are able to use one decision engine, and thus one joint policy set, for answering access queries of multiple access control enforcement points (e.g., for web access and Java method calls), we are not aware that this concept has ever been used with the multi-layer concept of data trees. In the long run, we believe that multi-layer setting of access policies is superior to only auditing. However, it requires exclusive write access for the new centralized product to all the policies of the individual systems, while multi-layer auditing only requires read access. Thus short- and mid-term, multi-layer auditing is much easier to realize.

References

1. Backes, M., Dürmuth, M., Steinwandt, R.: An algebra for composing enterprise privacy policies. In: Samarati, P., Ryan, P.Y A, Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 33–52. Springer, Heidelberg (2004)
2. Backes, M., Pfitzmann, B., Schunter, M.: A toolkit for managing enterprise privacy policies. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 162–180. Springer, Heidelberg (2003)
3. Bell, D.E., LaPadula, L.J.: Secure computer systems: Mathematical foundations. Technical Report 2547, Volume I, MITRE (1973) Available at: http://www.albany.edu/acc/courses/ia/classics/belllapadula1.pdf
4. Benaloh, J., Leichter, J.: Generalized secret sharing and monotone functions. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 27–36. Springer, Heidelberg (1990)
5. Bonatti, P.A., de Capitani di Vimercati, S., Samarati, P.: An algebra for composing access control policies. ACM Transactions on Information and System Security 5(1), 1–35 (2002)
6. Buecker, A., Watanabe, Y.: Design considerations for privacy-preserving database access. IBM Redbooks Paper (2003), http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/redp3720.
7. Castano, S., Fugini, M.G., Martella, G., Samarati, P.: Database Security. ACM Press, New York (1995)
8. De Capitani di Vimercat, S., Samarati, P.: An authorization model for federated systems. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) ESORICS 1996. LNCS, vol. 1146, pp. 99–117. Springer, Heidelberg (1996)
9. Denning, D.E., Denning, P.J.: Certification of programs for secure information flow. Communications of the ACM 20(7), 504–513 (1977)
10. Dennis, J.B., Horn, E.C.V.: Programming semantics for multiprogrammed computations. Communications of the ACM 9(3), 143–155 (1966)
11. Fischer-Hübner, S. (ed.): IT-Security and Privacy. LNCS, vol. 1958. Springer, Heidelberg (2001)
12. Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in operating systems. Communications of the ACM 19(8), 461–471 (1976)
13. Hosmer, H.: The multipolicy paradigm for trusted systems. In: Proc. ACM Workshop on New Security Paradigms, pp. 19–32. ACM Press, New York (1993)

14. Jones, A.K., Lipton, R.J., Snyder, L.: A linear time algorithm for deciding security. In: Proc. 17th IEEE FOCS, pp. 33–41. IEEE Computer Society Press, Los Alamitos (1976)
15. Karjoth, G., Schunter, M., Waidner, M.: The 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)
16. Kudo, M., Hada, S.: XML document security based on provisional authorization. In: Proc. 7th ACM CCS, pp. 87–96. ACM Press, New York (2000)
17. Li, N., Tripunitara, M.V.: On safety in discretionary access control. In: Proc. 26th IEEE Symp. on Security & Privacy, pp. 96–109. IEEE Computer Society Press, Los Alamitos (2005)
18. Moffett, J.D., Sloman, M.S.: Policy hierarchies for distributed systems management. Journal on Selected Areas in Communications 11(9), 1404–1414 (1993)
19. Popek, G.J.: Protection structures. Computer, pp. 22–33 (July 1974)
20. Rits, M., De Boe, B., Schaad, A.: XacT: A bridge between resource management and access control in multi-layered applications. In: ACM Workshop on Software Engineering for Secure Systems (SESS '05), pp. 1–7. ACM, New York (2005)
21. Saltzer, J.H.: Protection and the control of information sharing in Multics. Communications of the ACM 17(7), 388–402 (1974)
22. Sandhu, R.S.: The typed access matrix model. In: Proc. 13th IEEE Symp. on Security & Privacy, pp. 122–136. IEEE Computer Society Press, Los Alamitos (2002)
23. Wulf, W., Cohen, E., Corwin, W., Jones, A., Levin, R., Pierson, C., Pollack, F.: HYDRA: The kernel of a multiprocessor operating system. Communications of the ACM 17, 337–345 (1974)

Refinement for Administrative Policies

M.A.C. Dekker[1,2] and S. Etalle[2]

[1] Security group, TNO ICT, The Netherlands
[2] Distributed and Embedded Systems group, University of Twente, The Netherlands

Abstract. Flexibility of management is an important requisite for access control systems as it allows users to adapt the access control system in accordance with practical requirements. This paper builds on earlier work where we defined administrative policies for a general class of RBAC models. We present a formal definition of administrative refinement and we show that there is an ordering for administrative privileges which yields administrative refinements of policies. We argue (by giving an example) that this privilege ordering can be very useful in practice, and we prove that the privilege ordering is tractable.

1 Introduction

Role-based access control (RBAC) [1] is a well-known standard for access control, aimed to make the assignment of users to privileges more easy. In practice however, for example in hospitals or enterprises, RBAC policies can be very large and dynamic, consisting of thousands of roles [6], and changing frequently. In such cases *policy management* can be a daunting task. The usual approach to this problem is to divide the work and to delegate (bits of) administrative authority to other users. The advantage is that users can adapt the access control policy to changing circumstances more easily, without an administrative bottleneck. Not only does this reduce the cost of maintaining the access control policy, it also avoids bad security practices, such as sharing passwords or keys that should really remain secret. For example, it may be convenient to allow the head nurse to delegate database access to other nurses when they need it for particular tasks, without having to recur to the hospital's security officer. On the other hand, this kind of flexibility also introduces security risks as changes made to the RBAC policy could entail privacy breaches.

The issue of designing *flexible* yet *safe* policy administration mechanisms for RBAC has received much attention recently [3,4,6,9,14]. To mention some of the research: In ARBAC [9] *administrative privileges* are assigned to a separate hierarchy of *administrative roles* and defined by specifying a range of roles that can be changed. Crampton and Loizou [4] take a more general approach, by using the same hierarchy for both administrative privileges and ordinary user privileges. Using the concept of *administrative scope*, they define which roles should have administrative privileges over other roles. In a similar approach, Wang and Osborn [12] divide the role-graph (a type of RBAC policy) into administrative domains. Each administrative domain has one administrator with privileges about the (roles in the) domain. In the Role-Control Center [6], administrative privileges over roles are defined in terms of *views*, which are subsets of the role-hierarchy, and they can only be assigned to users that are assigned to these roles. There seems to be no consensus (yet) about which administrative privileges belong to which roles; each of the above mentioned frameworks differs on this issue. Some models motivate their choice by considerations that include the meaning of a *role* in a company, or the concepts of ownership, and responsibility, as one would find it in a company. On the other hand, Li et al. argue that interpreting the RBAC role hierarchy as a business organization chart can be misleading [8].

This paper aims to be a contribution to the above-mentioned lines of research on management of RBAC policies. In this paper we introduce the concept of *administrative refinement*, and we show that this concept yields a more flexible, and at the same time more safe administrative model. This paper builds on earlier work [5], where we argued informally that there is a natural ordering for administrative privileges. In this paper we present a formal definition of administrative refinement, and we show that it yields an ordering on the administrative privileges, which allows for a more flexible policy management. Furthermore, we present the formal proof that this privilege ordering is tractable. Note that in this paper we do not assume any features that go beyond the *General Hierarchical RBAC model* (such as constraints), and that we do not restrict which administrative privileges can be assigned to which roles. We are hence led to believe that our results are also applicable to a range of more advanced RBAC models.

2 Preliminaries

We first introduce shortly the *General Hierarchical RBAC* model, as defined in the ANSI RBAC standard, because it is the most commonly used RBAC model [1,6]. In Section 3 we extend this model with administrative privileges, yielding a general class of administrative policies.

The goal of an RBAC policy is to specify which users are permitted to perform which actions on which objects. We denote the sets of users, roles, actions, and objects, by U, R, A, and O. Permissions for performing actions on objects are called *user privileges*, forming a set $P \subseteq A \times O$, e.g. $(\mathit{read}, \mathit{ehrtable})$, $(\mathit{print}, \mathit{colorA4})$.

A non-administrative RBAC policy assigns users to roles, roles to user privileges, and it defines an order on the roles; the *role-hierarchy*[1].

Definition 1 (Non-administrative Policies). *Let U, R, and P be sets of users, roles, and user privileges,* a non-administrative RBAC policy ϕ *is a tuple*

$$\phi = (UA, RH, PA),$$

[1] In the RBAC standard the relation RH is defined to be acyclic, reflexive and transitive, i.e. it is defined to be a *partial order*. Li et al., however, showed that this definition causes problems when changes are made to the role-hierarchy [8]. Here, for the sake of generality we do not assume that RH is a partial order.

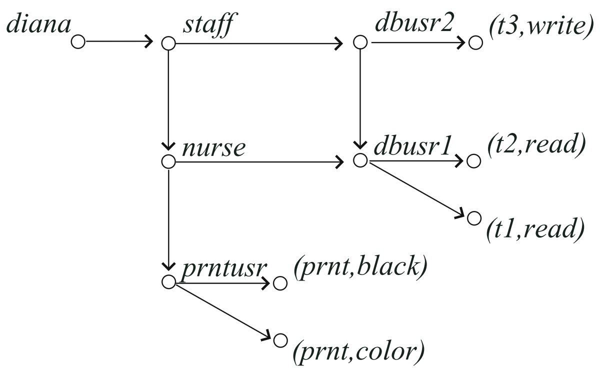


Fig. 1. Sample Non-Administrative RBAC policy

where $UA \subseteq U \times R$ determines which users are member of which roles, $RH \subseteq R \times R$ is the role-hierarchy, and $PA \subseteq R \times P$ determines which roles have which privileges.

The set of non-administrative RBAC policies is denoted $\Phi_{U,R,P}$. To simplify our exposition we treat a policy ϕ as a *directed graph*, defined by the *set* of directed edges $UA \cup RH \cup PA$. If there is a path from one vertex v to another v' we write $v \rightarrow_\phi v'$. Below we sometimes omit the subscript ϕ when the policy is clear from the context.

The RBAC *reference monitor* uses the policy ϕ as follows. Any user u can start a *session*. The reference monitor allows the user to *activate* a role r in a session iff $u \rightarrow_\phi r$. The privileges of the user's session are all the privileges p such that $r \rightarrow_\phi p$ for some role r activated in the session. Sessions are an important safety mechanism, allowing users to apply the *principle of least privilege*. Here, for the sake of simplicity we ignore the details about sessions. For details about sessions we refer the reader to the ANSI RBAC standard [1]. Let us give a simple example of a non-administrative RBAC policy.

Example 1 (Basic RBAC). Consider the setting of a hospital, where a database system *dbms* stores electronic health records in a number of tables *t1, t2, t3* etc. The health records can only be seen or changed by authorized personnel. To enforce this the system *dbms* uses the RBAC policy depicted in Figure 1: The employee *Diana* can activate the role *nurse* or the role *staff*. In the former case she can *read* the tables *t1* and *t2*, while in the latter case she can also *write* the table *t3*.

3 Administrative RBAC Policies

The RBAC standard specifies a number of *administrative functions and controls*, which can be used by an administrative authority to make policy changes [1].

In this paper we express administrative authority in terms of administrative privileges to model which users (or roles) can make which policy changes. There are two types of privileges: privileges for making new edges (denoted here by $\Box$), and privileges for removing edges (denoted here by $\Diamond$). We assign the administrative privileges to roles just like the user privileges are assigned to roles in standard RBAC. This approach is also advocated in the literature[2] and the intuition behind it is that the RBAC policy can also be used to specify who can change the RBAC policy [4,12].

Clearly, administrative privileges must be an *infinite* set, even if we assume that the sets of users, roles and user privileges are *finite*. The reason is that administrative privileges over administrative privileges are also administrative privileges. For example, consider the privilege to give someone else the privilege to change the members of a role. The number of *administrative levels* (the number of nestings of the $\Box$ connective to be introduced below) is often restricted in existing literature (sometimes to one [10] or to two levels [14]). We agree that in some settings multiple levels of administration are not useful, however, here we prefer to take a general approach, leaving it up to security officers to choose which administrative privileges to use in their systems.

We formalize the full set of privileges by defining a *grammar* that encompasses both user privileges and administrative privileges.

Definition 2 (Privilege Grammar). *Let U, R, P be sets of users, roles and user privileges, the set of all privileges $P^\dagger_{U,R,P}$ is defined by the following grammar:*

$$p \ ::= \ q \mid \Box(u,r) \mid \Diamond(u,r) \mid \Box(r,r') \mid \Diamond(r,r') \mid \Box(r,p) \mid \Diamond(r,p).$$

where $u \in U$, $r, r' \in R$, and $q \in P$.

Each administrative privilege corresponds to an administrative action in a straightforward way. For example, the privilege $\Box(u,r)$ allows to add a member u to the role r. The privilege $\Diamond(u,r)$ allows to remove a member u from the role r. For simplicity we do not model privileges to change the sets U, R, or P, and we assume that they are chosen sufficiently large and fixed. The rationale is that changes to U, R, or P do not actually change the policy, rather they change which policies are well-formed. For example, in practice the set of users could be chosen to be *all strings starting with 'uid'*, which is independent of which users are assigned to roles in the RBAC policy.

As mentioned, $\Box$ and $\Diamond$ are connectives and the set $P^\dagger$ is infinite. (Even if U, R, P are finite.) For example, one could have an expression $\Box(r, \Box(u, r'))$, which expresses the privilege to give to role r, the privilege to a user u to the role r'. We can now define administrative policies.

[2] Existing literature focusses however on defining constraints on which roles can have which administrative privileges. For example, to prevent low roles from obtaining privileges about higher roles [4]. For the sake of generality we do not make choices with respect to such constraints.

Definition 3 (Administrative Policies). *Let U, R, P be sets of users, roles and user privileges,* an administrative RBAC policy *ϕ is a tuple*

$$\phi = (UA,\ RH,\ PA^{\dagger}),$$

where $UA \subseteq U \times R$ is a set of user assignments, $RH \subseteq R \times R$ a role-hierarchy, and $PA^{\dagger} \subseteq R \times P^{\dagger}$ are the assignments to user or administrtaive privileges.

The set of administrative policies is denoted $\Phi^{\dagger}_{U,R,P}$, which is a superset of the policy set $\Phi_{U,R,P}$ from standard RBAC (see Definition 1). Administrative policies allow users to make policy changes. We model this formally by defining administrative commands.

Definition 4 (Administrative Commands). *Let U, R, P, be sets of users, roles and user privileges, an* administrative command *is a term*

$$\texttt{cmd}(u, a, v, v'),$$

where $u \in U$, $a \in \{\Box, \Diamond\}$ and $v, v' \in U \cup R \cup P^{\dagger}$.
A command queue *is a list of administrative commands, denoted cq = $\texttt{cmd}(u, a, v_1, v_2) : \texttt{cmd}(u', a', v'_1, v'_2)...$, where $:$ denotes the list constructor.*

The set of command queues is denoted CQ. The empty command queue is denoted ε. The administrative functionality of the RBAC reference monitor is modeled by a command queue, and an administrative RBAC policy. The RBAC reference monitor changes the policy by executing administrative commands in the command queue. We formalize this by a transition function.

Definition 5 (Administrative Transition). *Let $cq \in CQ$ be a command queue, and $\phi \in \Phi^{\dagger}$ an administrative policy, the* administrative transition function*, denoted $\Rightarrow$:$\Rightarrow$: $CQ \times \Phi^{\dagger} \rightarrow CQ \times \Phi^{\dagger}$, is*

$$\begin{aligned}
\langle \texttt{cmd}(u, \Box, v, v') : cq,\ \phi\rangle &\Rightarrow \langle cq,\ \phi \cup (v, v')\rangle, \textit{ if } u \rightarrow_{\phi} r \textit{ and } r \rightarrow_{\phi} \Box(v, v'). \\
\langle \texttt{cmd}(u, \Diamond, v, v') : cq,\ \phi\rangle &\Rightarrow \langle cq,\ \phi \setminus (v, v')\rangle, \textit{ if } u \rightarrow_{\phi} r \textit{ and } r \rightarrow_{\phi} \Diamond(v, v'). \\
\langle \texttt{cmd}(\ldots) : cq,\ \phi\rangle &\Rightarrow \langle cq,\ \phi\rangle, \textit{ otherwise.}
\end{aligned}$$

Note that if an administrative command is not allowed by the policy ϕ, then the command is removed from the queue, without changing the policy ϕ. Below, a sequence of executions of commands in the queue is called a *run*, denoted by $\Rightarrow^{*}$. We give a brief example by applying this model in a practical situation.

Example 2. Consider the policy in Example 1. Alice, the security officer, wants to delegate some of her administrative authority to the employees of the Human

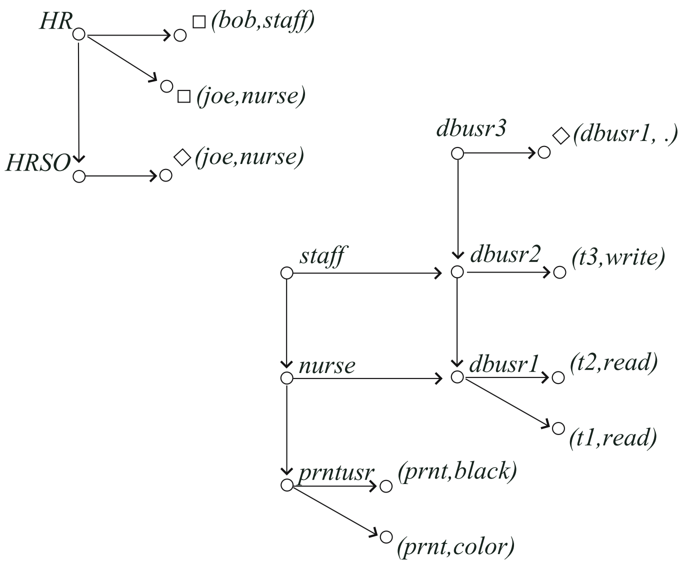


Fig. 2. The administrative RBAC policy deployed by Alice, the security officer

Resource department (HR). In this way, members of *HR* can appoint new staff members or nurses, without having to recur to Alice each time. To delegate these administrative privileges, Alice uses an administrative policy.

Figure 2 shows Alice's policy: Members of *HR* can assign and revoke certain users to *staff* and *nurse* roles. Additionally, to protect the confidentiality of health records in the tables *t2* and *t3* Alice delegated a revocation privilege about the role *dbusr2* to the role *dbusr3*. The administrative policy hence not only describes who can access which resources, but also which roles have privileges to change to the policy.

4 Administrative Refinement

In the previous section we have defined a general class of administrative policies for the General Hierachical RBAC model. In existing literature [3,4,6,9,14], the administrative privileges in RBAC policies are treated just like ordinary user privileges. In this section we show that this is more restrictive than necessary for safety, and that a more flexible approach can be very useful in practice. This section is organized as follows. First we formalize the notion of *administrative refinement*. In section 4.1 we show that the privilege ordering for *assignment privileges* [5] yields administrative refinements of policies, and in section 4.2 we present the formal proof that the privilege ordering is decidable.

Ignoring policy changes for the moment, an access control policy ψ is safer than a policy ϕ, if ψ grants users to less privileges than ϕ does. We call this *non-administrative refinement.*

Definition 6 (Non-Administrative Refinement). *Let $\phi, \psi \in \Phi^\dagger$ be two RBAC policies. We say that ψ is a non-administrative refinement of ϕ, denoted $\phi \succeq \psi$, iff for any $v \in U \cup R$ and any* user privilege *$p \in P$, $v \rightarrow_\psi p$ implies $v \rightarrow_\phi p$.*

We give a basic example to illustrate this definition.

Example 3 (Non-Administrative Refinement). Consider the policy depicted in Figure [1]. Clearly, by removing any of the edges in the policy one obtains a refinement of the policy. For example, by removing Diana from the *staff* role. There is a more fine-grained type of refinement that rearranges edges. For example, if we replace the edge between *Diana* and *staff* with an edge between *Diana* and *nurse*, then we have another refinement of the policy. On the other hand, if we replace the edge between *nurse* and *dbusr1* with an edge between *nurse* and *dbusr2*, we do not obtain a refinement, as nurses get more privileges.

We can now define administrative refinement. The goal of an administrative policy is to allow certain policy changes. Basically, an administrative refinement of a policy is a policy that allows *safer* policy changes. Note that a policy change made by one user may allow other users to make new policy changes, and so on. Therefore, to determine the possible policy changes that are allowed, we must take into account which users are performing administrative actions, and in which order[3]. We formalize administrative refinement as follows.

Definition 7 (Administrative Refinement). *Let $\phi, \psi \in \Phi^\dagger$ be administrative RBAC policies. We say that ψ is an* administrative refinement *of ϕ, denoted $\phi \succeq^\dagger \psi$, if, for any queue of administrative commands $cq \in CQ$, there is a queue of administrative commands $cq' \in CQ$, such that $\phi' \succeq \psi'$, where $\langle cq, \phi \rangle \Rightarrow^* \langle \varepsilon, \phi' \rangle$, and $\langle cq', \psi \rangle \Rightarrow^* \langle \varepsilon, \psi' \rangle$, and cq' is such that, it contains the same number of commands, and the n-th command in cq and the n-th command in cq' are both of the form $\texttt{cmd}(u, ., .)$, where n ranges over the number of commands in the queue cq.*

Basically the definition states that, if ψ allows a certain policy change then either the same policy change is also allowed by the policy ϕ, or it is a policy change that results in a *safer* policy. It is easy to see that administrative refinement implies non-administrative refinement; take $cq = cq' = \varepsilon$. In other words, if $\phi \succeq^\dagger \psi$ holds then also $\phi \succeq \psi$ holds.

[3] Taking into account the order is more precise than in the HRU model [7] where it is assumed that there is a group of untrusted users who can collude in any order, which does not allow to distinguish the policy *lowrole* $\rightarrow \square(r, p)$ from *highrole* $\rightarrow \square(r, p)$ (but the latter is more safe).

4.1 Ordering Administrative Privileges

In this section, we introduce first a privilege ordering on administrative privileges [5] and we show that the ordering of the administrative privileges yields administrative refinements of a policy. At the end of this section we show how the privilege ordering can be used in practice to allow more flexible policy management.

Consider a simple setting where a sub-administrator has the explicit right to assign a user u to a *high* role in the role-hierarchy. There is no reason to forbid the sub-administrator to assign the user to a lower role. This can be seen as follows. If u becomes a member of the high role, then u can activate also the lower roles and obtain their privileges, as if u was assigned to it explicitly. In existing RBAC literature administrative privileges are not interpreted in this way. The ordering of privileges, just described here, can be defined formally as follows.

Definition 8 (Privilege Ordering). *Let $\phi \in \Phi^\dagger$ be an administrative policy, let p, p_1, p_2 be privileges in $P^\dagger$, and let v_1, v_2, v_3, v_4 be users (U) or roles (R). We define the relation $\rightsquigarrow_\phi$ as the smallest relation satisfying:*

$$p \rightsquigarrow_\phi p \tag{1}$$

$$\Box(v_2, v_3) \rightsquigarrow_\phi \Box(v_1, v_4), \textit{ if } \; v_1 \rightarrow_\phi v_2 \textit{ and } v_3 \rightarrow_\phi v_4. \tag{2}$$

$$\Box(v_2, p_1) \rightsquigarrow_\phi \Box(v_1, p_2), \textit{ if } \; v_1 \rightarrow_\phi v_2 \textit{ and } p_1 \rightsquigarrow_\phi p_2. \tag{3}$$

The ordering $\rightsquigarrow_\phi$ is both reflexive and transitive. In practice the privilege ordering can be used to allow users, with administrative privileges, to be implicitly authorized for weaker administrative privileges.

It can be shown (see the Theorem 1 below) that by replacing an administrative privilege by a weaker one (with respect to the ordering), one obtains an administrative refinement of the policy. In other words, giving administrative users also the weaker administrative privileges allows them to perform also *safer* administrative operations than the ones originally allowed.

Theorem 1. *Let $\phi \in \Phi^\dagger$ be an administrative policy, let $(r, p) \in \phi$ be a privilege assignment, and let q be a privilege such that $p \rightsquigarrow_\phi q$, then the policy $\psi = (\phi \setminus (r, p)) \cup (r, q)$ is an administrative refinement of ϕ, that is $\phi \succeq^\dagger \psi$.*

Proof. (Sketch) The proof is by case analysis over the different cases in definition 8.
The first case is trivial, since the relation $\succeq^\dagger$ is reflexive. For the second case take a policy ϕ with privilege assignment $(r, \Box(v_2, v_3))$, and $v_1 \rightarrow_\phi v_2$, and $v_3 \rightarrow_\phi v_4$. Let ψ be the same policy where this privilege assignment is replaced by $(r, \Box(v_1, v_4))$. So ϕ allows the command

$$\texttt{cmd}(u, \Box, v_2, v_3),$$

which changes ϕ to $\phi' = \phi \cup (v_2, v_3)$, while ψ allows the command

$$\texttt{cmd}(u, \Box, v_1, v_4),$$

which changes ψ to $\psi' = \psi \cup (v_2, v_3)$. To show that $\phi \succeq^\dagger \psi$ it is sufficient to show that $\phi' \succeq \psi$: In ψ' v_1 has the privileges of $v4$, but in ϕ' v_1 has the same privileges, due to the edges $v_1 \rightarrow_\phi v_2$, $v_3 \rightarrow_\phi v_4$ and $v_2 \rightarrow v_3$.

For the third case take a policy ϕ with privilege assignment $(r, \Box(v_2, p_1))$, and $v_1 \rightarrow_\phi v_2$, and $p_1 \rightsquigarrow_\phi p_2$. Let ψ be the same policy where this privilege assignment is replaced by the weaker privilege $(r, \Box(v_1, p_2))$. So ϕ allows the command

$$\texttt{cmd}(u, \Box, v_2, p_1),$$

which changes ϕ to $\phi' = \phi \cup (v_2, p_1)$, while ψ allows the command

$$\texttt{cmd}(u, \Box, v_1, p_2),$$

which changes ψ to $\psi' = \psi \cup (v_1, p_2)$. In case p_1 is a user privilege, p_1 equals p_2 and the proof is the same as for the second case. We simply show that ψ' is a non-administrative refinement of ϕ'. On the other hand, if p_1 is an administrative privilege we must show that the subsequent commands allowed by ψ' yield refinements of the policies created by commands allowed by ϕ'. This can be shown by induction over the structure (the number of nestings of $\Box$) of p_1.

Let us now give an example of how the privilege ordering can be used in a practical situation.

Example 4 (A Flexworker). Consider the administrative RBAC policy depicted in Figure 2. The role *HR* has the administrative privilege to add new members to the staff role. There is also a role below staff called nurse, with additional privileges. Bob is a flexworker, Jane is from the *HR* department.

Bob arrives at the hospital and his job is to put some order in the health record database. To do the job he needs to have *dbusr2* privileges. Jane a member of the role *HR* can give the necessary clearance to Bob. Jane can give Bob staff privileges (the dashed edge in Figure 3). If she does so, then she must urge Bob to apply the principle of least privilege, by activating only the role *dbusr2*, and not e.g. the staff or the nurse role, which would yield excessive privileges, for instance medical privileges. But Jane can only hope that Bob does so.

The privilege ordering implies that Jane can assign Bob directly to the *dbusr2* role (the dotted edge in Figure 3) *because* of her privilege to add Bob to the staff role. In a way, instead of preaching the principle of least privilege to Bob, Jane applies it for him.

Remark 1 (Less privileges, safer policies). In this paper we have defined a policy to be safer when the policy gives users less privileges. The principle of least privilege, and the way it is supported by the RBAC session mechanism, is a well-known example of the usefulness of this definition. One could perhaps argue that there could be practical situations where having less privileges is not more

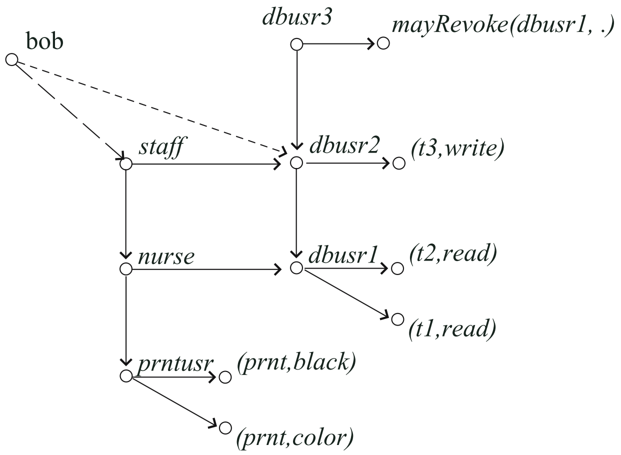


Fig. 3. A practical example of the use of administrative refinement

safe. For example one could imagine a privilege to append to a log file. Removing this privilege could cause programs to run unsafely, that is without writing logs. We believe however that such peculiarities should be resolved at the application layer. For example by changing the program so that it halts when no logs can be written.

4.2 Tractability

In this section we address an important practical issue. We prove that the ordering relation (Definition 8) is tractable. Since the full set P of privileges is infinite, this result is not immediate. For instance, a naive forward search does not necessarily terminate (see the example at the end of this section). The proof indicates how a decision algorithm, deciding which privileges are to be given to which roles, can be implemented at an RBAC reference monitor.

Lemma 1 (Decidability of the Ordering Relation). *Let $\phi \in \Phi^\dagger$ be an administrative policy, and $p, q \in P^\dagger$ be two privileges, it is decidable whether $p \rightsquigarrow_\phi q$.*

Proof. The proof is by structural induction over q.

The base cases are when q is not of the form $\Box(r, r')$. We show that for the three base cases $p \rightsquigarrow_\phi q$ is decidable:

– Either q is a user privilege from P. In this case $p \rightsquigarrow_\phi q$ holds only when $p = q$ (see rule (1) in Definition 8).
– Or q is of the form $\Box(v, v')$ for some $v, v' \in U \cup R$, in which case only rule (2) needs to be checked, which has finite premises.

For the induction step, suppose that q is $\Box(r', p')$, for some role r' and privilege p'. Now, $p \rightsquigarrow q$ can only hold if the premises of rule (3) hold. The premises of rule (3) are decidable, either because they are finite, or because the induction hypothesis is applicable (in $p' \rightsquigarrow q'$, q' is structurally smaller than q, regardless of p').

Let us show how the proof above can be used in practice, as a procedure for checking whether one privilege is weaker than another.

Example 5. Consider Example 4 again. Can Jane assign Bob to the *dbusr2* role? We have to check that the role staff inherits the privilege $\Box(bob, dbusr2)$. Using the first part of Definition 8, one finds that the staff role has the privilege $\Box(bob, staff)$. Now we should decide whether

$$\Box(bob, staff) \rightsquigarrow \Box(bob, dbusr2).$$

This follows trivially from the first rule of Definition 8.

To give a more involved example, suppose that the system administrator Alice has the privilege $\Box(staff, \Box(bob, staff))$. Can Charlie give to staff, the privilege $\Box(bob, dbusr2)$ directly? We have to check whether

$$\Box(staff, \Box(bob, staff)) \rightsquigarrow \Box(staff, \Box(bob, dbusr2)).$$

This is indeed the case by using rule (3) first, and then rule (2).

Now, for the sake of exposition, let us remove the edge between the *staff* and the *dbusr2* role. Let us show how to determine that the previous relation does not hold: Now only rule (3) applies, in which case we must decide whether $\Box(bob, staff) \rightsquigarrow \Box(bob, dbusr2)$. This is a base case of the induction described in the proof of Lemma 1: Only rule (2) remains to be checked and than we can conclude that it does not hold.

It could be useful to find *all* the privileges p' weaker than a given p. To our surprise, in some cases the set of all privileges p' weaker than a given privilege p, is infinite. Let us give an example.

Example 6 (Infinitely many weaker privileges). Consider a policy where $(r_2, \Box(r_1, r_2)) \in PA$. We should stress here that this is by no means an artificial, or peculiar policy: Members of r_2 can make members of r_1 member too.

Suppose we are interested in finding all the privileges weaker than $\Box(r_1, r_2)$. The first weaker privilege we discover by applying rule (2) in definition 8:

$$\Box(r_1, \Box(r_1, r_2)).$$

Using this result in rule (3), we find another weaker privilege,

$$\Box(r_1, \Box(r_1, \Box(r_1, r_2))),$$

and we can use this again in rule (3), and so on.

Remark 2. The outer nesting in the last term in the previous example is in a sense redundant. Instead of assigning the privilege $\Box(r_1, r_2)$ to r_1, one assigns the privilege to do so, to r_1. This only requires the users in role r_1 to perform an extra administrative step, which seems unnecessary. It is cumbersome for the user in r_1, and it does not introduce any safeguards. We conjecture that for all practical purpose one could stop after n applications of rule (3), where n is the length of the longest chain in RH. We do not make this observation more formal here.

5 Related Work

The problem of administration of an RBAC system was first addressed by Sandhu et al. [10]. Later, numerous articles have been published extending or improving the administration model proposed there [3,4,6,9,11,13,14]. We discuss some of them.

Barka et al. [3] distinguish between original and delegated user role assignments. Delegations are modeled using special sets, and different sets are used for single step and double step delegations (which must remain disjoint). A function is used to verify if membership to a role can be delegated. Privileges can also be delegated, provided they are in the special set of delegatable privileges belonging to the role. In their work, each level of delegation requires the definition of tens of sets and functions, whereas in our model administrative privileges, of an arbitrary complexity, are simply assigned to roles, just like the ordinary privileges. The PDBM model [14] defines a cascaded delegation. This form of delegation is also expressible in our grammar (by nesting the $\Box$ connective). In the PDBM model, however, each delegation requires the addition of a separate role, which is cumbersome given the fact that there are already many roles to manage. In our model the administrative privileges are assigned to roles just like the ordinary privileges. It is not required to add any additional roles.

A number of proposals define general constraints on the administrative privileges. For example, the constraint that a user must first have a privilege, before being allowed to delegate it to other users. Note that, as mentioned earlier, in this paper no particular choice is made with respect to such constraints. Zhang et al. [13] implement rule based constraints on delegations. They demonstrate their model using a Prolog program. Basically, they analyze the properties of a centralized RBAC system, focussing on so-called *separation of duty* policies. Crampton [4] defines the concept of administrative scope. Basically a role r is in the scope of a role r' if there is no role above r' that is not below r. They show how administrative scope can be used to constrain delegations to evolve in a natural progression in the role hierarchy. Bandman et al. [2] use a general constraint language to specify constraints on who can receive certain delegations.

6 Conclusion

With this work we make a contribution to the design of flexible administration models for RBAC. Flexible administration is important to cut the cost of maintenance and to enable the RBAC system to adapt to changing circumstances. In general, the flexibility of management is a very important requisite for access control systems. Discretionary access control systems are prevalently used (see for instance Linux, Windows) because users can change the policies about their files so easily. Mandatory access control systems, on the other hand (such as RBAC) are deployed to a lesser extent because they are too inflexible. There are settings where flexibility is required, but discretionary access control is inappropriate. A good example is the setting of the protection of electronic health records. The high availability requirements for health records require flexibility, and at the same time, policies protecting health records are not at the discretion of medical personnel creating and using them. RBAC, with a flexible decentralized policy management mechanism could be an interesting solution in such settings.

The issue of designing *flexible* yet *safe* policy administration mechanisms for RBAC has received much attention recently [3,4,6,9,14]. With this paper we contribute to these lines of research. We introduce the notion of administrative refinement of policies, and we show how it can be used to allow more flexible management of the RBAC policy. Concretely, our contribution is a the definition of a general class of administrative policies, and a formal definition of administrative refinement. We have shown that there is a natural ordering for administrative privileges which yields administrative refinements of policies, and we have presented the proof that this ordering is tractable. We also showed how useful our extension is in practice. Our approach allows administrative users to be implicitly authorized for weaker administrative operations, which is thus more flexible and more safe as well.

Revocation privileges are included in our model, but we have not identified (yet) a separate ordering for revocation privileges. We believe that this is an interesting possibility for further research.

References

1. Standard, R.B.A.C.: ANSI INCITS 359-2004 (2004)
2. Bandmann, O.L., Sadighi Firozabadi, B., Dam, M.: Constrained delegation. In: Abadi, M., Bellovin, S.M. (eds.) Proc. of the Symp. on Security and Privacy (S&P), pp. 131–140. IEEE Computer Society Press, Los Alamitos (2002)
3. Barka, E., Sandhu, R.S.: Framework for role-based delegation models. In: Epstein, J., Notargiacomo, L., Anderson, R. (eds.) Annual Computer Security Applications Conference (ACSAC), pp. 168–176 (2000)
4. Crampton, J., Loizou, G.: Administrative scope: A foundation for role-based administrative models. Transactions on Information System Security (TISSEC) 6(2), 201–231 (2003)
5. Dekker, M.A.C., Cederquist, J., Crampton, J., Etalle, S.: Extended privilege inheritance in RBAC. In: Proc. of the Symp. on Information, Computer and Communications Security (ASIACCS), ACM Press, New York (2007) (to be published)
6. Ferraiolo, D.F., Kuhn, D.R., Chandramouli, R.: Role-based Access Control. Computer Security Series. Artech House (2003)
7. Harrison, M.A, Ruzzo, W.L., Ullman, J.D.: Protection in operating systems. Comunications of the ACM 19(5) (1976)
8. Li, N., Byun, J., Bertino, E.: A critique of the ANSI standard on role based access control. IEEE Security and Privacy (page in press)
9. Sandhu, R.S., Bhamidipati, V., Munawer, Q.: The ARBAC97 model for role-based administration of roles. Transactions on Information and System Security (TISSEC) 2(1), 105–135 (1999)
10. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)

11. Wainer, J., Kumar, A.: A fine-grained, controllable, user-to-user delegation method in RBAC. In: Ferrari, E., Ahn, G. (eds.) Proc. of the Symp. on Access Control Models and Technologies (SACMAT), pp. 59–66. ACM Press, New York (2005)
12. Wang, H., Osborn, S.L.: An administrative model for role graphs. In: Proc. of the IFIP TC-11 WG 11, pp. 302–315. Kluwer Academic Publishers, Dordrecht (2003)
13. Zhang, L., Ahn, G., Chu, B.: A rule-based framework for role-based delegation and revocation. Transactions on Information and System Security (TISSEC) 6(3), 404–441 (2003)
14. Zhang, X., Oh, S., Sandhu, R.S.: PBDM: a flexible delegation model in RBAC. In: Ferraiolo, D. (ed.) Proc. of the Symp. on Access Control Models and Technologies (SACMAT), pp. 149–157. ACM Press, New York (2003)

Authenticating kNN Query Results in Data Publishing

Weiwei Cheng and Kian-Lee Tan

Department of Computer Science
National University of Singapore
{chengwei,tankl}@comp.nus.edu.sg

Abstract. In data publishing model, data owners engage third-party data publishers to manage their data and process queries on their behalf. As the publishers may be untrusted or susceptible to attacks, it could produce incorrect query results. In this paper, we extend the signature-based mechanism for users to verify that their answers for k nearest neighbors queries on a multidimensional dataset are complete (i.e. no qualifying data points are omitted), authentic (i.e. no answer points are tampered) and minimal (i.e. no non-answer points are returned in the plain). Essentially, our scheme returns k answer points in the plain, and a set of $(\tilde{p}, q)$-pairs, where $\tilde{p}$ is the digest of a non-answer point p in the dataset used to facilitate the signature chaining mechanism to verify the authenticity of the answer points, and q is a reference point (not in the dataset) used to verify that p is indeed further away from the query point than the kth nearest point. We study two instantiations of the approach - one based on the native data space using space partitioning method (a.k.a. R-tree) and the other based on the metric space using iDistance. We conducted an experimental study, and report our findings here.

1 Introduction

In data publishing model, data owners engage third-party data publishers to manage their data and process queries on their behalf [6,10]. This model is applicable to a wide range of computing platforms, including database caching [8], content delivery network [19], edge computing [9], P2P database [7], etc.

The data publishing model offers several advantages over conventional client-server architecture where the owner undertakes the processing of user queries. First, network latency could be reduced by pushing application logic and data processing from the owner's data center out to multiple publisher servers situated near user clusters. Second, it would be cheaper to achieve scalability by adding publisher servers than fortifying the owner's data center and provisioning more bandwidth for every user. Third, the data publishing model removes the single point of failure in the owner's data center, hence reducing the database's susceptibility to denial of service attacks and improving service availability.

In this paper, our primary concern is the threat that a publisher may return incorrect query results to the users, whether intentionally or under the influence

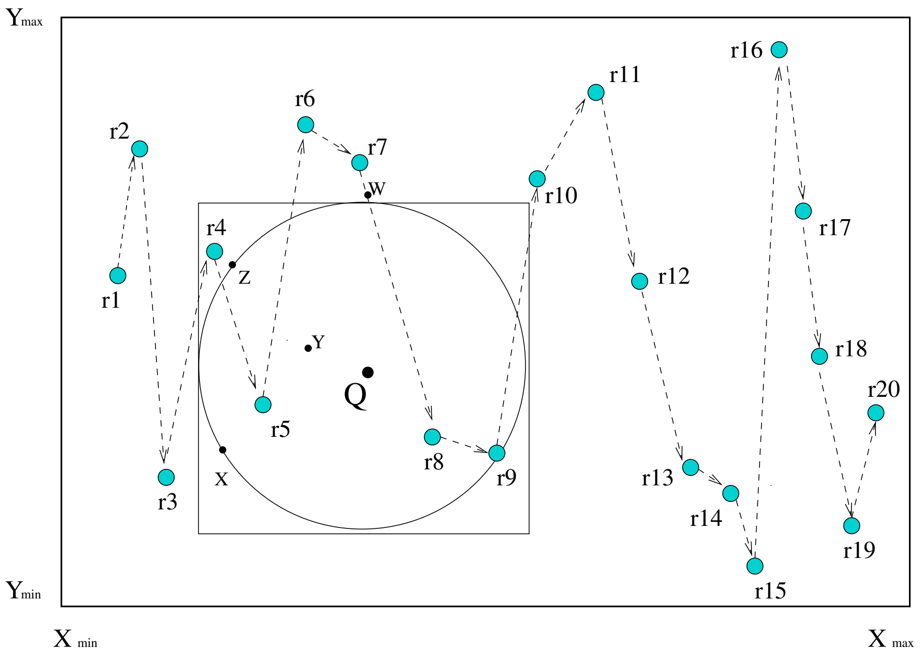


Fig. 1. A Running Example

of an adversary. An adversary who is cognizant of the data organization in the publisher server may make logical alterations to the data, thus inducing incorrect query results. In addition, a compromised publisher server can be made to return incomplete query results by withholding data intentionally. Therefore mechanisms for users to verify the completeness as well as authenticity of their query results are essential for data publishing model. Moreover, it is highly desirable that only answers are returned in the plain to facilitate confidentiality and access control.

Several existing works provide for checking the authenticity [11,14] and completeness [6,13] of query results. However, most of them only deal with one-dimensional datasets. Devanbu's scheme [6] handles multiple key attributes by essentially concatenating them in some preferred order $key_1|key_2|...|key_n$; this scheme is expected to be very inefficient for symmetric queries, such as window and nearest neighbor queries, that are typical in multi-dimensional context. In our previous paper [5], we described a partition-based query authentication mechanism for multi-dimensional database; however, the mechanism is designed for authenticating results of hyper-rectangle window queries. While this scheme can be used for kNN queries, it will return more than k points in the plain, and thus may violate confidentiality and access control policies.

In this paper, we propose a mechanism for users to verify the answer of a k nearest neighbors (kNN) query on a multi-dimensional dataset. Like existing works [5,13], our mechanism is based on the signature chain concept, and verifies that the k NN answers are complete (i.e. no qualifying data points are omitted), authentic (i.e. no answer points are tampered) and minimal (i.e. no non-answer points are returned in the plain). The core of the scheme is to return k answer points in the plain, and a set of $(\tilde{p}, q)$-pairs, where $\tilde{p}$ is the digest of a non-answer point p in the dataset used to facilitate the signature chaining mechanism to verify the authenticity of the answer points, and q is a reference point (not in the dataset) used to verify that p is indeed further away from the query point than the kth nearest point. The scheme is minimal since only the k answer points are revealed in the plain. We study two instantiations of the approach - one based on the native data space using space partitioning method (a.k.a. R-tree) and the other based on the metric space using iDistance. We have implemented both techniques, and our results show that the R-tree-based scheme has better performance when the number of dimensions is low ($d < 8$), while iDistance-based scheme is superior in high-dimensional datasets ($d > 8$). To our knowledge, this is the first reported work that addresses this problem.

The remainder of this paper is organized as follows: Section 2 presents some background and gives an overview of our proposed method. In Sections 3 and 4, we present how to handle kNN queries under the native and metric space respectively. Section 5 presents results from a performance study. Finally, Section 6 concludes the paper.

2 The Big Picture

The general setting of our problem is as follows. A data owner of a multi-dimensional dataset $\mathcal{DB}$ outsourced the management of $\mathcal{DB}$ to a third-party publisher. Besides $\mathcal{DB}$, (s)he also created one or several associated signatures of $\mathcal{DB}$ that are outsourced together with it. Users are also made aware of certain meta-data, as well as the public key of the owner. During kNN query processing, the publisher returns k answers and the associated *verification objects* (VOs) for the users to verify the correctness of the answers.

We note that there are other security issues that the data publishing model poses such as privacy, user authentication and access control. These have been studied extensively (e.g. [1], [15], [12], [18]), and are orthogonal to our work here.

2.1 Problem Definition

Figure 1 shows a 2-dimensional data space comprising 20 data points, r_1 to r_{20}. The query Q [1] is a 3NN query (i.e., $k = 3$). As shown, the query answer of this example is $\{r_5, r_8, r_9\}$, where r_9 is the 3rd NN (furthest among the 3 answers from Q). Let $dist(x,y)$ denote the distance between two points x and y. The circle shown in the figure, whose center is Q and radius $dist(Q,r_9)$, is the hyper-sphere range query of Q. The square, whose center is Q and length $2 \cdot dist(Q, r_9)$, is the corresponding most tightly bounded hyper-cube window query of Q.

[1] In our discussion, we often overload Q to refer to three related terms: kNN query, the query point, and the center of the hyper-sphere. The context should be clear from the text.

To process a kNN query Q, the server performs two tasks. First, it evaluates Q and returns a result set R. This task is basically the traditional kNN query processing and hence any kNN query evaluation strategy can be adopted. Second, it constructs verification objects (VOs) for the result set R. In this paper, we focus on the problem of constructing VOs that the user will use to verify the correctness of R.

We note that for kNN queries, the user knows the number of answer points (which is k). As such, the server must return k points. For the adversary to cheat, (s)he can (a) tamper with the data (e.g., replace content of point r_5); (b) drop a true kNN point and replace it with a fictitious point (e.g., replace point r_5 by the fictitious point Y within the same hyper-sphere about Q); (c) drop a true kNN point and replace it with another point that is further away from Q than the kth NN point (e.g., drop r_5 and return r_4 expanding the radius of the hyper-sphere of Q to $dist(Q, r_4)$ which is longer than $dist(Q, r_9)$). Cases (a) and (b) can be easily handled by techniques that verify the authenticity of a point - in fact, our signature scheme can deal with this easily. The challenge is to handle case (c) to ensure that the k points returned are indeed the k NN points.

From the user's perspective, (s)he knows the k answer points in R that are returned. In addition, (s)he can construct a hyper-sphere about Q whose radius is $dist(Q, z)$ where z is the kth NN point in R. In this way, we could view the kNN query authentication problem as the problem to ensure (a) all the k NN points returned have not been tampered with, and (b) no other points in the dataset has distance shorter than $dist(Q, z)$.

2.2 Background

Before we present our solutions, there are three important concepts that are critical to our work. We briefly review them here. We assume a d-dimension data space. Let $L = (L_1, L_2, \ldots, L_d)$ and $U = (U_1, U_2, \ldots, U_d)$ be two points that bound the entire d-dimensional data space, where $L_r \leq U_r$ for all r. L and U are known to all users. Suppose the space contains N data points $p_1 = (x_{11}, x_{12}, \ldots, x_{1d})$, ... $p_N = (x_{N1}, x_{N2}, \ldots, x_{Nd})$.

Signature Chain. Each data point in the database has associated with it a signature, which is obtained based on the signature chain method [13,5]. Under the signature chain scheme, the signature of a point is derived from a combination of the point itself and its left and right neighboring points. Points can be ordered arbitrarily - this will not affect the correctness of the proposed scheme but the efficiency. Without loss of generality, we assume that p_i is ordered before p_j for $1 \leq i < j \leq N$. Moreover, L is ordered before p_1 and U is ordered after p_N. Once the points are ordered, the signature of a point p_i, $sig(p_i)$, is given as follows:[2]

$$sig(p_i) = s(h(g(p_{i-1})|g(p_i)|g(p_{i+1}))) \quad (1)$$

[2] For the end points, the bounding points are used in the signature computation. Due to space constraint, we will not discuss this. See [5] for details.

where s is a signature function using the owner's private key (e.g., RSA [17] and DSA [2]), h and g are one-way hash functions (e.g., MD5 [16] and SHA [3]), and $|$ denotes concatenation. Referring to our example in Figure 1, points are ordered based on the x-dimension and so r_1 is ordered before r_2, which is ordered before r_3 and so on. The signatures of these points are thus "chained" as shown by the dotted arrow lines. In other words, the signature of r_5 is computed from points r_4, r_5 and r_6.

Collaborative Computation of Digest. The signature of a point is dependent on the use of one-way hash function g used to compute the digest of a point. In this work, we adopt an iterative one-way hash function of the form [5]:

$$g(p_i) = \sum_{r=1}^{d} h^{U_r - x_{ir} - 1}(x_{ir}) | h^{x_{ir} - L_r - 1}(x_{ir}) \tag{2}$$

where $h^j(x_{ir}) = h^{j-1}(h(x_{ir}))$ and $h^0(x_{ir})$ applies a one-way hash function on x.

We note that this iterative hash function facilitates the user and server to collaboratively determine the digest of a point p. The basic idea is that given a *reference* point q known to both the user and the server, the server can partially compute the digest of p wrt q and then the user completes the computation wrt q. To illustrate, let a point $p = \{x_1, x_2, ..., x_d\}$ and another point $q = \{y_1, y_2, ..., y_d\}$, such that $x_i < y_i \; \forall i$. Then, instead of returning the digest of p directly, the server can compute $h^{y_i - x_i - 1}(x_i)$ and $h^{x_i - L_i - 1}(x_i)$. The user will then derive $g(p)$ using Equation 2 after applying h on $(h^{y_i - x_i - 1}(x_i))$ an additional of $(U_i - y_i)$ times to get $(h^{U_i - x_i - 1}(x_i)) \; \forall i$. Now, similar computation can be derived for different relations between x_i and y_i.

Hiding Non-Answer Points. The combination of the above two concepts - signature chain and collaborative computation - provides a very powerful mechanism to hide non-answer points, i.e., while the digest of non-answer points can be returned, their actual values are not known to the user. In this way, the server cannot tamper with the original digest (since the digest is used in the signature computation), while the confidentiality of data is preserved and access control is not violated. As an example, consider three points p, r, t which form a signature chain. Suppose r and t are answer points while p is not. Moreover, suppose there is reference point q. Now we can verify that r is correct by verifying that its signature (returned as a verification object) is correct without returning p in the plain. This can be done since we can compute the digest of p wrt q collaboratively by the server and user, and then compute the signature of r based on Equation 1. Clearly, if the server attempts to cheat on p, then the user will not be able to obtain the right digest for p. Thus, non-answer points can be hidden.

Challenges. The key challenge for authenticating kNN queries is to identify the reference points. In [13,5], the reference points are finite and known - in [13], the user-specified boundary of a query range has only at most two values (e.g., return all answers such that $100 < x < 200$; here, 100 and 200 are the boudary points and they are given); in [5], the reference points are the two bounding points of an MBR which is well defined. For kNN queries, while we can bound the answers (with the hyper-sphere), we have essentially an infinite number of boundary points (all points on the hyper-sphere!). Thus, we need another mechanism to pick and control the number of reference points.

2.3 The Basic Solution

Our proposed solution, in its most basic form, ensures authenticity, completeness, and minimality, and works as follows. Once the server computes the kNN answers, it returns only the k answers in plaintext. In addition, it also returns the following verification objects:

– It returns the k signatures of the answer points. These are used to verify that the data have not been tampered with.
– The k points returned may not fall into a consecutive sequence along the signature chain. For example, in Figure 1, there is a gap between r_5 and r_8. Thus, the server will also need to return the partial computation of the digests of a number of points that form a chain. Referring to our example again, we need to return the partial digests of points r_3, r_4, r_6, r_7 and r_{10}. We will defer the discussion on how these points are determined to the later sections when we present the respective techniques. (It suffices at this moment to note that we must return r_3 to be certain that there is no point within the hyper-sphere that is chained between r_3 and r_4.) The user will then derive the digests of these points to verify the authenticity of the answer points. For example, by computing the digests of r_4 and r_6, we can verify if r_5 is authentic. Similarly, with the digest of r_7, we can verify if r_8 is authentic. Similarly, the digest of r_{10} is needed to verify the authenticity of r_9.
– Now, for the user to verify that the answers are indeed the k answer points, it need to show that all other points in the chain are outside of the hyper-sphere centered at Q with radius given by the distance between Q and the kth answer point. Using our example, the user need to verify that r_3, r_4, r_6, r_7 and r_{10} are outside of the hyper-sphere. To do this, the server also returns a set of reference points. Let the number of non-answer points returned be M. Then, the number of reference points needed is M, one for each of the non-answer points. These reference points are points in the space but not from the dataset. Moreover, they are points on or outside of the hyper-sphere surface so that the distance between these points and Q is larger than the radius of the hyper-sphere, but shorter than the distance between their corresponding non-answer points and Q. Note that the server can easily determined these since the server knows all the points. Using our running example again, r_3 has a reference point X. For each (non-answer point, reference point) pair, the partial digest of the non-answer point is computed by the server (as described in Section 2.2), and the user can complete the computation and derive the actual digest of the non-answer point. As long as the digest is valid, the user will know that the non-answer point is outside of the hyper-sphere (since it

knows that the distance between Q and the reference point is larger than the radius of the hyper-sphere). We will discuss how the reference points are selected in subsequent sections (since not any arbitrary reference point works). In addition, we note that we can optimize the number of reference points returned since it is possible that a number of non-answer points can use the same reference point. Referring to our example, one reference point W can be used for both points r_6 and r_7.

Taking our example in figure 1 again, the query answer for this 3NN query Q is $\{r_5, r_8, r_9\}$. Besides the plaintext for this 3 answers, the server also returns the following verification objects:

– Signatures of the 3 answer points, which are $sig(r_5)$, $sig(r_8)$ and $sig(r_9)$.
– For the two boundary points r_3 and r_{10} of the answer's signature chain returned, the server returns two pairs $(\tilde{r_3}, B_1)$ and $(\tilde{r_{10}}, B_2)$, where $\tilde{r_3}$ and $\tilde{r_{10}}$ are the partial computation of the digests of r_3 and r_{10} respectively. Points B_1 and B_2 are the leftmost and rightmost point of the hyper-sphere query respectively, where $B_1.x = Q.x - dist(Q, r_9)$ and $B_2.x = Q.x + dist(Q, r_9)$.
– For points r_4, r_6, and r_7 that fall into the gap of the answer points along the consecutive signature chain sequence, the server returns pairs $(\tilde{r_4}, Z)$, $(\tilde{r_6}, W)$, and $(\tilde{r_7}, W)$ respectively, where $\tilde{r_i}$ is the partial digest of point r_i, Z and W are the corresponding reference points selected for each r_i.

Clearly, the proposed method is minimal since only the k answer points are returned in the plain!

3 kNN Authentication in Native Space

In this section, we present our solution for authenticating kNN queries in the native space. We shall adopt the data partitioning approach in our discussion. Figure 2 shows an example of how the dataset in Figure 1 has been partitioned. We shall use this figure for illustration.

Given a data space D, let $L = (L_1, L_2, \ldots, L_d)$ and $U = (U_1, U_2, \ldots, U_d)$ be two points that bound the entire data space, where $L_i \leq U_i$ for all i. L and U are known to all users. Let Q be a kNN query. We denote the hyper-sphere formed by the answers of Q as $Q(O, r)$, where O is the center point and $r = dist(O, kth\ NN\ answer\ point)$ is the radius of the hyper-sphere.

Consider a partition P bounded by two points $p_0 = (x_{01}, x_{02}, \ldots, x_{0d})$ and $p_{n+1} = (x_{(n+1),1}, x_{(n+1),2}, \ldots, x_{(n+1),d})$ where $x_{0z} \leq x_{(n+1),z}$ for all z. Suppose P contains n data points $p_1 = (x_{11}, x_{12}, \ldots, x_{1d})$, $\ldots$ $p_n = (x_{n1}, x_{n2}, \ldots, x_{nd})$. Without loss of generality, we assume that points are ordered based on increasing value of the first dimension. Moreover, we assume $x_{11} \leq x_{21} \leq \ldots \leq x_{n1}$. Thus, p_i is ordered before p_j for $1 \leq i < j \leq n$. Clearly, p_0 is ordered before p_1 and p_{n+1} is ordered after p_n. In our example, the points in partition R1 are ordered as r_1, r_2, r_4, while that in partition R4 are ordered as r_{12}, r_{16}, r_{17}. Furthermore, each

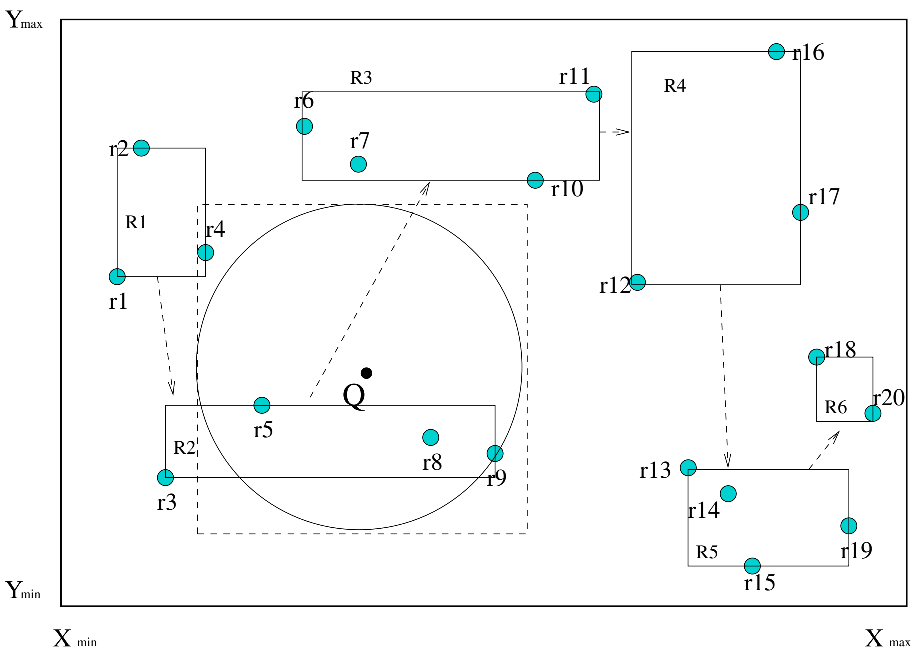


Fig. 2. R-tree-based kNN Query Authentication

partition P has a signature derived from its bounding points and the number of points it contains:

$$sig(P) = s(h(g(p_0)|g(p_{n+1})|h(n))) \tag{3}$$

Finally, partitions are also ordered (based on the value of the first dimension), and signature-chained. Referring to Figure 2, we have R1 ordered/chained before R2, and so on.

Such an approach facilitates pruning of the space – unlike Figure 1 where we have to examine 8 points (r_3 to r_{10}), as shown in Figure 2, we now need to examine only the 4 points in partition R2. However, the verification process becomes more complex and involves two steps now: a) we need to verify that none of the valid partitions have been missed; b) for partitions that should be checked, none of the valid points have been missed.

Verifying that the query answer covers all the candidate partitions is straightforward for a known hyper-sphere. It comprises the following two phases:

– In the first phase, we need to identify the list of candidate partitions. A partition is a candidate if the range of the ordering dimension of its MBR overlaps the range of the ordering dimension of the tightest hyper-cube that bounds the hyper-sphere. In our example, only partitions R1, R2 and R3 are candidate partitions.
– In the second phase, some of these candidate partitions can be further pruned. The ones to be pruned are those whose minimum distance to O is

larger than the hyper-sphere radius (since no points inside these partitions will ever be nearer than the kth NN answer point. This can be easily verified since the bounding points of each partition are known. In our example, R1 and R3 are further pruned.

For each remaining candidate partition P, there are 3 possible relationships between P and the hyper-sphere $Q(O, r)$:

1. $Q(O, r)$ contains P. In this case, we return all the points in P, i.e., the server returns p_0 to p_{n+1} and n, together with the respective signatures $sig(P_0)$ to $sig(P_{n+1})$ and $sig(P)$. The user would compute the digests for both the points and the partition to verify the result.
2. P contains $Q(O, r)$. Let $P_i = (x_{i1}, x_{i2}, ..., x_{id})$, and $O = (o_1, o_2, ..., o_d)$. Let Q' be the most tightly bounded hyper-cube of Q, thus Q' is also centered at point O, and the length of each edge $l = 2r$. Let Q''s bounding points be $(q_{l1}, q_{l2}, ..., q_{ld})$ and $(q_{u1}, q_{u2}, ..., q_{ud})$. Thus, $q_{ui} = O_i + r$ and $q_{li} = O_i - r$ for all $i \in [1, d]$. The data points in P can be separated into
   (a) $p_\alpha, p_{\alpha+1}, ..., p_{\beta-1}, p_\beta$, such that $x_{i1} \in [q_{l1}, q_{u1}]$ for $\alpha \leq i \leq \beta$.
   These points can be further categorized into answer points ($\mathcal{A}$) and false positives ($\mathcal{F}$). For each answer point $p_i \in \mathcal{A}$, $dist(O, P_i) \leq r$, and for each false positive $p_i \in \mathcal{F}$, $dist(O, P_i) > r$. Furthermore, there are two types of false positive points. In the first type, denoted $\mathcal{F}_a$, for each $p_i \in \mathcal{F}_a$, $\exists z, x_{iz} \notin [q_{lz}, q_{uz}]$. In the second type, denoted $\mathcal{F}_b$, for each $p_i \in \mathcal{F}_b$, $\forall z, x_{iz} \in [q_{lz}, q_{uz}]$. Note that $\mathcal{F}_a$ corresponds to points outside the hyper-cube, while $\mathcal{F}_b$ are points inside the hyper-cube but outside the hyper-sphere. Let us use the data space in Figure 1 as an example of a partition containing the hyper-sphere. Here, we have $\mathcal{A} = \{r_5, r_8, r_9\}$, $\mathcal{F}_a = \{r_6, r_7\}$ and $\mathcal{F}_b = \{r_4\}$.
   (b) $p_1, ...p_{\alpha-1}, p_{\beta+1}, ...p_k$, which are clearly not answer points. Referring to Figure 1, these points are r_1 to r_3 and r_{10} to r_{20}.

   For data points from different categories, the server returns different sets of verification objects.
   (a) For each point $p_i \in \mathcal{A}$, the server returns p_i and $sig(p_i)$.
   (b) The server also returns p_0, p_{n+1}, $sig(p_0)$ and $sig(p_{n+1})$, and $sig(P)$.
   (c) For each point $p_i \in \mathcal{F}_a \cup \mathcal{F}_b \cup \{p_{\alpha-1}, p_{\beta+1}\}$[3], the server finds a reference point $S = (S_1, S_2, ..., S_d)$ on the surface of the hyper-sphere[4], such that, if $x_{iz} < o_z$, $S_z \in (x_{iz}, o_z)$, else if $x_{iz} > o_z$, $S_z \in (o_z, x_{iz})$.

[3] For points in $\mathcal{F}_a \cup \{p_{\alpha-1}, p_{\beta+1}\}$, we can apply the technique in [5] to verify that these points are outside the hyper-cube Q' by treating the hyper-cube Q' as a window query. In this case, there is no need to transmit any reference point. However, for uniformity in discussion, and to keep the presentation simple, we just discuss the proposed approach.

[4] We do not require the point to be on the surface. All that is needed is to find a point that is outside of the hypersphere that is closer to the query point than the point to be hidden. However, for ease of presentation, we shall refer to the reference point as a point on the surface.

We note that the same S point could be used as a reference point for multiple p_is as long as the above conditions hold. For simplicity, we pick the point closest to the sphere's surface on the line joining O and p_i. Among these points, we then eliminate "redundant" reference points.
After an S point is chosen for each $p_i \in \mathcal{F}_b$, we could simply verify that $dist(O, p_i) > dist(O, S) \geq r$.
The server then returns several pieces of information together with the detailed information of point S:
 i. if $x_{iz} < S_z$, $h^{S_z - x_{iz} - 1}(x_{iz})$ and $h^{x_{iz} - L_z - 1}(x_{iz})$ are returned.
 ii. if $x_{iz} > S_z$, $h^{U_z - x_{iz} - 1}(x_{iz})$ and $h^{x_{iz} - S_z - 1}(x_{iz})$ are returned.

With the above information, the user can compute $g(p_i)$ without knowing the actual value of p_i.
- if $x_{iz} < S_z$, the user applies h on $h^{S_z - x_{iz} - 1}(x_{iz})$ an additional $(U_z - S_z)$ times to get $h^{U_z - x_{iz} - 1}(x_{iz})$.
- if $x_{iz} > S_z$, the user applies h on $h^{x_{iz} - S_z - 1}(x_{iz})$ an additional $(S_z - L_z)$ times to get $h^{x_{iz} - L_z - 1}(x_{iz})$.
- The user computes $g(p_i)$ using Equation 1.

Consider Figure 1 again as our example where P contains $Q(O, r)$. We could see that the point r_7 is outside the hyper-cube, which means that r_7 is not an answer of Q. Instead of just returning the value of r_7, the server picks a reference point W near the circle, where $W.x > r_7.x$ and $W.y < r_7.y$. Then (part of the information) the server returns: for query answers $\{r_8, r_9\}$, it returns r_8, r_9, $sig(r_8)$, and $sig(r_9)$; for r_7, it returns (1) $h^{W.x - r_7.x - 1}(r_7.x)$ and $h^{r_7.x - L.x - 1}(r_7.x)$;(2) $h^{U.y - r_7.y - 1}(r_7.y)$ and $h^{r_7.y - W.y - 1}(r_7.y)$. Here, L and U denote the two bounding points of the partition. With these, the user can determine $h^{U.x - r_7.x - 1}(r_7.x)$ and $h^{r_7.y - L.y - 1}(r_7.y)$, and compute the digest of r_7. (S)he can then further verify that r_8 is an answer point.

3. P overlaps $Q(O, r)$. This case can be handled by splitting P into two parts: one overlaps Q' (the hyper-cube of $Q(O, r)$), and the other does not overlap Q' (which means it does not overlap $Q(O, r)$). For the first part, we handle it in the same manner as case (2) above. For the second part, it can be dropped (except to verify that its points are outside Q'). As such, we shall not go into the details of this case.

In the above discussion, we have assumed only one layer of partitioning. We can easily extend the scheme to work with data structures like R-tree, where the data space is recursively partitioned, with internal nodes covering a larger space. In this case, all that is needed is to further chain the MBRs of each internal node to verify that no internal nodes are tampered with and dropped unnecessarily.

4 kNN Authentication in Metric Space

In Section 3, we have looked at how to authenticate kNN queries in the native data space. In this section, we shall look at the problem when points are stored in the metric space. Many data structures have been designed for processing

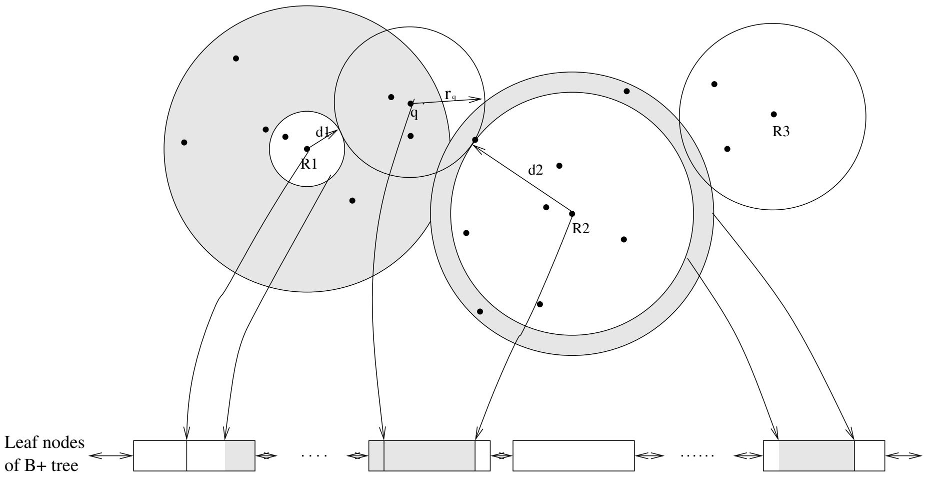


Fig. 3. iDistance based scheme

kNN queries in metric space. We shall discuss the method that is based on the iDistance [20] scheme here.

iDistance is an efficient technique for kNN search that can be adapted to different data distributions. In iDistance, the data space is partitioned according to a set of reference points. By indexing the distance of each data point to the reference point of its partition, high-dimensional points are transformed into points in a single dimensional space and indexed by a classical B+-tree. In particular, points in a partition are mapped into a range of values in the single dimensional space such that no two partitions have overlapping ranges. Thus, all points in partition P_i is located to the left side of points in partition P_{i+1} in the B+-tree.[5] Within the same partition, data points are ordered by their distance from the data point to its reference point. Referring to Figure 3, we have 3 partitions formed by 3 reference points R1, R2 and R3 respectively. A range query with center at q and radius r will need to access data points in the shaded region shown in the figure.

In the iDistance scheme, data partitioning is independent of the spatial location of the data points but only related to the selection of reference points. Moreover, the shape of a partition P_j in the iDistance structure is a hyper-sphere that is centered at its reference point O_j with radius $r_{P_j} = max(dist(r_i, O_j))$. Let a hyper-sphere query be centered at Q with radius r_q. Partition P_j does not overlap with the query and can be pruned from further consideration if the following holds:

$$dist(Q, O_j) \geq r_{P_j} + r_q \tag{4}$$

[5] We note that the original iDistance scheme did not discuss how partitions are ordered. Here, we adopt a simple strategy that orders the partition based on the values of the first dimension of the reference point.

On the other hand, if $dist(Q, O_j) < r_{P_j} + r_q$, we have to return the detailed information to show that all the query results contained in this partition are returned correctly. Now, as reported in [20], the set of points that need to be examined are bounded by the following inequality

$$dist(Q, O_j) - r_q \leq dist(O_j, r_i) \leq dist(Q, O_j) + r_q \quad (5)$$

In the authentication model, we build up the signature chain directly on top of the B+-tree. Let $O_j = (O_{j1}, O_{j2}, ..., O_{jd})$ be the reference point for partition P_j. The signature of each data point r_i is

$$sig(r_i) = s(g(r_{i-1})|g(r_i)|g(r_{i+1})) \quad (6)$$

where $g(r_i) = h(h(r_i)|h(dist(r_i, O_j)))$. Moreover, for each partition P_j,

$$sig(P_j) = s(h(O_j)|h(max(dist(r_i, O_j)))|h(k)) \quad (7)$$

where $h(O_j) = h(h(O_{j1})|h(O_{j2})| \ldots |h(O_{jd}))$ and k is the number of data points contained in partition P_j.

Like the R-tree based scheme, authentication of kNN queries for the iDistance based scheme contains the following two steps: (a) Verify that no overlapped partitions is missed out; (b) Verify that no result points inside the overlapped partition is tampered or dropped.

To verify that all overlapped partitions are returned, the publisher need to return the following information to the client:

– For each partition P_j, return O_j, r_{P_j}, k and $sig(P_j)$. With these information, the client can verify that the partition information has not been tampered with. Moreover, the client can safely prune away partitions that satisfy Equation 4 from further verification.

Here, we assume that the client knows the number of partitions; otherwise, additional information has to be provided (e.g., the signature for the total number of partitions, and the number of partitions). We note that this phase can be optimized by chaining the partitions to minimize the amount of information to be sent to the client. This is similar to the process of verifying partitions in the R-tree based scheme.

Now, for each partition P_j that overlaps the query hyper-sphere, we need to verify that no points has been tampered or dropped. The publisher returns the following information to facilitate verification:

– The continuous sequence of signature chain within P_j that satify Equation 5. Since the signatures are ordered by the distance to the reference point, those points matching the inequality would form a continuous signature chain and should be returned to the user as verification objects. Since not all points with the same distance are answer points, this chain of points contain both answer points $\mathcal{A}$ and false positives $\mathcal{F}$. For each point $p_i \in \mathcal{A}$, the publisher returns p_i and $sig(p_i)$. For each point $p_j \in \mathcal{F}$, the publisher returns a reference point $S = (S_1, S_2, ..., S_d)$ on the hyper-sphere (in the native space) as well as the corresponding (partial) digest. As in the R-tree based scheme, different false positive points could share a same reference point S as long as the following condition holds: $if\ r_{iz} < Q_z, S_z \in (r_{iz}, Q_z);\ else\ S_z \in (Q_z, r_{iz})$, $1 \leq z \leq d$.

– The publisher also returns the (partial) digests of the two points bounding the continuous sequence of signature chain above. Essentially, these two points allow the client to verify that no other points within the partition has been dropped. Each of these points is also associated with a reference point.

We note that the verification process is done in the native space. Once the client receives all the verification objects, it operates in the native space in the same manner as that described in the R-tree based scheme. In other words, with the k answer points, it can determine the hyper-sphere query and hyper-cube query. For each of the non-answer points, the client uses its associated reference point to verify that it lies outside the hyper-sphere.

5 Performance Study

We have implemented the proposed solutions and conducted a series of experiments to study their performance. For the native space based scheme, we implemented the R*-tree data structure [4]. For the metric space scheme, we employed the iDistance scheme. The codes for both mechanisms are implemented in C++. The performance metrics used in our study is the authentication overhead introduced and the I/O access cost. The authentication overhead is computed as *the number of overhead points*/k, where the number of overhead points refer to the number of non-answer points returned.

Unless stated otherwise, we use the following default parameter settings. The number of dimensions is 4. The data distribution is Gaussian, the number of data points is 100K, the domain of each dimension is [0, 1M]. The node capacity is 30 (i.e., each node holds up to 30 data points). Queries are generated by randomly picking a point from the database, and the value of k for the kNN query is 10. For each experiment, we vary one of the above parameters, run 200 queries, and take the average score.

5.1 Effect of Number of Dimensions

We first vary the number of dimensions from 2 to 32. Figure 4 summarizes the result. As expected, a higher dimensionality introduces more overhead for both mechanisms adopted, as more non-answer points are required to verify the completeness of the query. Moreover, as the number of dimensions increases, the data space “expands” correspondingly; with a fixed dataset size, the data points for higher dimensional dataset are spread more sparsely. Thus, given kNN queries with the same k value, the radius of the corresponding hyper-sphere in a higher dimensional dataset is much larger than its radius in a lower dimensional dataset.

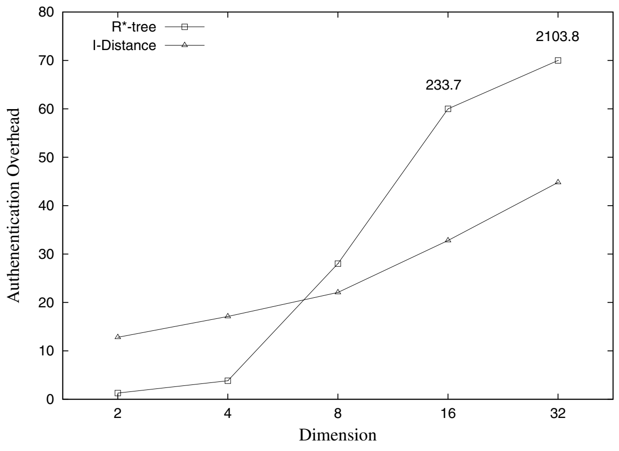


Fig. 4. Authentication Overhead on Different Data Dimension

Another observation is that for small number of dimensions, the R*-tree based mechanism yields lower authentication overhead. However, the iDistance based mechanism is superior when the number of dimensions is higher. This is reasonable as R*-tree has its own structural restriction when the dimensionality is high.

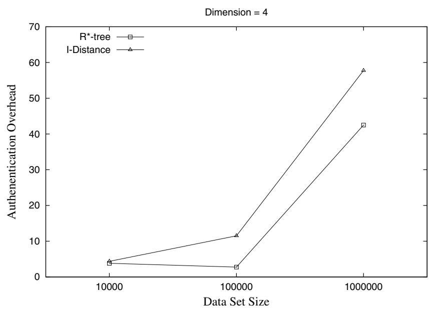
5.2 Effect of Different Dataset Size

In our second experiment, we study the effect of different dataset size for a fixed data space. Figure 5 shows the authentication overhead of the two schemes under different dataset size.

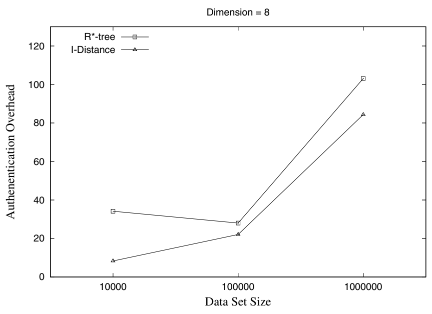
From the result, we observe that as the dataset size increases, the authentication overhead for iDistance based method increases as well. However, for the R*-tree based mechanism, the overhead decreases initially. Our investigation suggests the following reasons - the increasing dataset size reduces the size of the kNN query, which actually reduces the radius of its corresponding hyper-sphere. The R*-tree based method is more sensitive to this kind of reduction because of the overlaps in the MBR of its internal nodes in the structure. However, as the dataset size increases further, given the fixed data space, the space becomes too dense, resulting in larger overhead.

5.3 Effect of Different Data Distributions

In this experiment, we study the effect of different data distributions. As shown in figure 6, the results are measured under three different distribution: Exponential, Uniform and Gaussian. We note that both methods incur lesser overheads with the exponential dataset. This is because the data generated under the exponential distribution are clustered toward one corner (the origin) of the data space, whereas they are more spread out under the other two distributions. Moreover, the relative performance of the two methods remains the same for

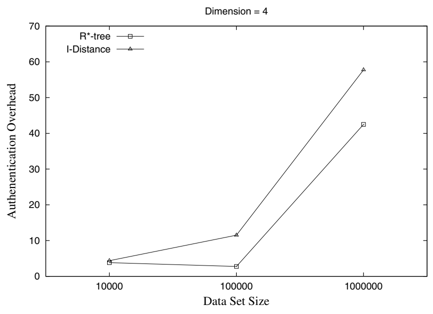


(a) d=4

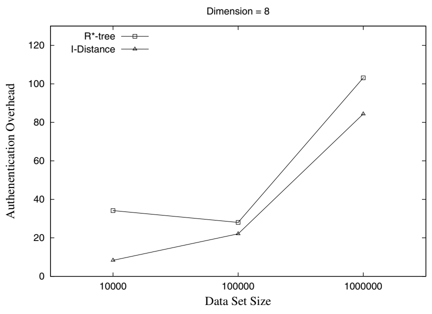


(b) d=8

Fig. 5. Authentication Overhead on different Dataset Size



(a) d=4



(b) d=8

Fig. 6. Authentication Overhead on different Data Distribution

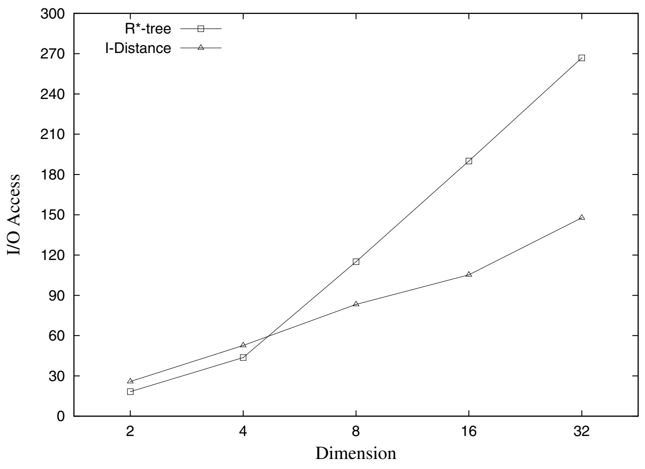


Fig. 7. I/O Access Cost

different data distributions. This result is also consistent with the findings in [5] for multi-dimensional window queries.

5.4 I/O Access Cost

Figure 7 shows the I/O access cost for the two mechanisms at the server. We see that the R*-tree based method outperforms the iDistance based method when the number of dimensions is small, while it incurs more I/O cost when the number of dimensions is large. This is consistent with previous works since the R*-tree method degenerates in performance as the number of dimensions increases.

6 Conclusion

In this paper, we have introduced a solution for user to verify their answers for kNN queries. This solution extends the multi-dimensional signature chain scheme by introducing a positional reference point P for each non-answer point examined. We studied two schemes, one is based on the R-tree structure and the other is based on the iDistance structure. Our experimental study showed that the method based on iDistance structure introduces less overhead and I/O access over the R-tree method for high dimensionality, while the R-tree based method is superior when the number of dimensions is small.

Acknowledgements

This work is partially supported by the university research grant R-252-000-228-112.

References

1. Encrypting File System (EFS) for Windows (2000), `http://www.microsoft.com/windows2000/techinfo/howit works/security/encrypt`
2. Proposed Federal Information Processing Standard for Digital Signature Standard (DSS). Federal Register 56(169), 42980–42982 (1991)
3. Secure Hashing Algorithm. National Institute of Science and Technology. FIPS 180-182 (2001)
4. Beckmann, N., Kriegel, H., Schneider, R., Seeger, B.: The r*-tree: An efficient and robust access method for points and rectangles. In: SIGMOD Conference, pp. 322–331 (1990)
5. Cheng, W., Pang, H., Tan, K.: Authenticating multi-dimensional query results in data publishing. In: Proceedings of the 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec'2006), pp. 60–73 (2006)
6. Devanbu, P., Gertz, M., Martel, C., Stubblebine, S.: Authentic Data Publication over the Internet. In: 14th IFIP 11.3 Working Conference in Database Security, pp. 102–112 (2000)
7. Huebsch, R., Hellerstein, J., Lanham, N., Loo, B., Shenker, S., Stoica, I.: Querying the Internet with PIER. In: Proceedings of the 29th International Conference on Very Large Databases, pp. 321–332 (2003)

8. Luo, Q., Krishnamurthy, S., Mohan, C., Pirahesh, H., Woo, H., Lindsay, B., Naughton, J.: Middle-Tier Database Caching for E-Business. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 600–611. ACM Press, New York (2002)
9. Margulius, D.: Apps. on the Edge. InfoWorld, 24(21), (May 2002), http://www.infoworld.com/article/02/05/23/020527feedgetci_1.html
10. Miklau, G., Suciu, D.: Controlling Access to Published Data Using Cryptography. In: Proceedings of the 29th International Conference on Very Large Data Bases, pp. 898–909 (2003)
11. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and Integrity in Outsourced Databases. In: Proceedings of the Network and Distributed System Security Symposium (February 2004)
12. Neuman, B., Tso, T.: Kerberos: An Authentication Service for Computer Networks. IEEE Communications Magazine 32(9), 33–38 (1994)
13. Pang, H., Jain, A., Ramamritham, K., Tan, K.: Verifying Completeness of Relational Query Results in Data Publishing. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, ACM Press, New York (2005)
14. Pang, H., Tan, K.: Authenticating Query Results in Edge Computing. In: Conference on Data Engineering, pp. 560–571. IEEE Computer Society Press, Los Alamitos (2004)
15. Pang, H., Tan, K., Zhou, X.: StegFS: A Steganographic File System. In: Proceedings of the 19th International Conference on Data Engineering, Bangalore, India, pp. 657–668 (March 2003)
16. Rivest, R.: RFC 1321: The MD5 Message-Digest Algorithm. Internet Activities Board (1992)
17. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
18. Sandhu, R., Samarati, P.: Access Control: Principles and Practice. IEEE Communications Magazine 32(9), 40–48 (1994)
19. Saroiu, S., Gummadi, K., Dunn, R., Gribble, S., Levy, H.: An Analysis of Internet Content Delivery Systems. In: Proceedings of the 5th Symposium on Operating Systems Design and Implementation, pp. 315–327 (2002)
20. Yu, C., Ooi, B., Tan, K., Jagadish, H.: Indexing the distance: An efficient method to knn processing. In: Proceedings of the 27th International Conference on Very Large Databases, pp. 421–430 (2001)

Query Rewriting Algorithm Evaluation for XML Security Views

Nataliya Rassadko

The University of Trento, via Sommarive 14, 38050 Povo(TN), Italy
rassadko@dit.unitn.it

Abstract. We investigate the experimental effectiveness of query rewriting over XML security views. Our model consists of access control policies specified over DTDs with XPath expression for data-dependent access control policies. We provide the notion of *security views* for characterizing information accessible to authorized users. This is a transformed (sanitized) DTD schema that is used by users for query formulation. To avoid the overhead of view materialization in query answering, these queries later undergo rewriting so that they are valid over the original DTD schema, and thus the query answer is computed from the original XML data. We provide an algorithm for query rewriting and show its performance compared with the naive approach, i.e. the approach that requires view materialization.

Keywords: query rewriting, XML views, XPath annotation, algorithm, evaluation, security.

1 Introduction

Specification of access control models for XML data has been a fairly active field of research in recent years [5], [6], [7], [10], [12], [13], [14], [16], [17], [19], [21], [22], [25], [27], [33]. All this previous work (except [14], [22], [27]) enforces security constraints at the document level by fully annotating the entire XML document. As a result, one major limitation of these models is the lack of support for authorized users to query the data: they either do not provide schema information of the accessible data, or expose the entire original DTD (or its so-called "loosened" variant). In both cases, the solution is hardly practical for large and complex documents. Furthermore, fixing the access control policies at the instance level without providing or computing a schema, makes it difficult for the security officer to understand how the authorized view of a document for a user or a class of users will actually look like. On the other side, revelation of excessive schema information might lead to security breaches: an unauthorized user can deduce or infer confidential information via multiple queries and analysis of the schema even if only the accessible nodes are queried.

To overcome this limitations, the notion of XML security views was initially proposed by Stoica and Farkas [33] and later refined by Fan et al. [14] and Kuper et al. [22]. The basic idea is to provide a schema that describes the data that can be seen by the user, as well as a (hidden) set of XPath expressions that describe how to compute the data in the view from the original data.

In the current paper, we implement and test experimentally the performance of the security view model of [22]. To this end, we define a rewriting algorithm that takes a user query over the a security view, and rewrites the query into a query over the original XML document. We then compare the cost of evaluating this query with that of evaluating the original query over a materialized view of the data, and show that significant performance improvements.

The paper is organized as follows. In Sec. 2 we present preliminary notions on XPath and XML security views. The algorithm of query rewriting is described shortly in Sec. 3. Evaluation of rewriting algorithm is provided in Sec. 4. The discussion of related work is located in Sec. 5. Finally, we conclude the paper in Sec. 6.

2 Background

We first review the fragment of XPath [11] that may be used by a user in query formulation.

Definition 1. *An XPath expression in $\mathcal{X}$ is defined by the following grammar:*

$$\begin{array}{rcl}
\langle xpath\rangle & ::= & `/`?\,\langle path\rangle \;\mid\; \langle path\rangle\,(`\cup`\,\langle path\rangle)\,* \\
\langle path\rangle & ::= & \langle step\rangle\,(`/`\,\langle step\rangle)\,* \\
\langle step\rangle & ::= & \langle test\rangle\,(`[`\,\langle qual\rangle\,`]`)\,* \\
\langle test\rangle & ::= & \theta`::`A \;\mid\; \theta`::*` \\
\langle qual\rangle & ::= & \langle xpath\rangle \;\mid\; \langle path\rangle\; op\; c \;\mid\; \langle qual\rangle\;\mathit{and}\;\langle qual\rangle \;\mid \\
 & & \langle qual\rangle\;\mathit{or}\;\langle qual\rangle \;\mid\; \mathit{not}\;\langle qual\rangle \;\mid\; `(`\,\langle qual\rangle\,`)`
\end{array}$$

where θ stands for an axis, c is a `str` *constant, A is a label, op stands for one of $=, <, >, \leq, \geq$. The result of the qual filtering is called* qualifier *and is denoted by q.*

We must note the semantic difference between $\langle xpath\rangle$ and $\langle path\rangle$: the former may contain unions, while the latter may not. This is because XPath 1.0 [11] does not allow unions in location steps. For the sake of readability, we ignore the syntactic difference between *xpath* and *path*; we denote both with p. We also abbreviate `self` by ϵ, `child` $:: A/p$ with A/p, `descendant-or-self` $:: A/p$ by $//A/p$, and $p = p_1/p_2$ with $p_2 = //p_2'$ is written p as $p_1//p_2'$. The parent axis is also abbreviated as ../.

The intuition behind XML security view is similar to that of multi-level security view for relational databases [24], [30], [32], [35], discretionary access control over relational databases [4] and object-oriented databases [8]. A typical approach to specifying and enforcing access control for traditional databases is to define views on which security permissions should be applied. For instance, Lunt et al. [23] showed how to use standard SQL queries to implement the SeaView multi-level secure database. In contrast, the hierarchical structure and the dependency (e.g., ancestor and descendants) of XML data as well as the presence of disjunction and recursion in DTDs make it impossible to define a security

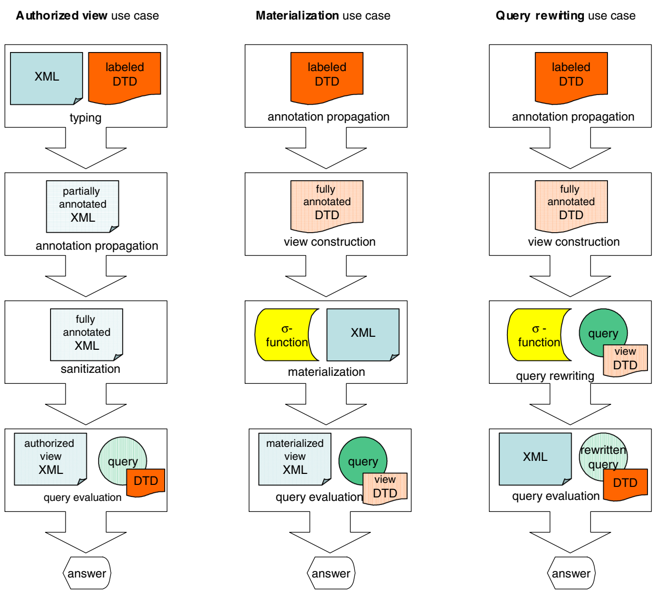


Fig. 1. Query answering use cases

view via a single query. Moreover, they introduce new challenges in how to generate a view that conforms to a DTD view. The challenges were observed in [3], which showed that even for XML views of relational data, it is highly nontrivial to ensure that the views typecheck.

The insight of XML security view construction was found in [30] where it was demonstrated how to compute a full database labelling from a partial one implied by security views. From the XML viewpoint, a partial assignment of security labels to XML document nodes can be also extended to a full assignment. From the latter, it is easy to compute an XML analogue of relational view by means of "sanitization" operation that hides (e.g., deletes or encrypts) nodes with negative authorizations, but reveals their permitted children. The resulting XML tree is called *authorized*. This scenario is depicted on the left part of Fig. 1.

Unlike solutions presented in e.g., [5], [13], [16], [21], we employ schema-level policy enforcement, resulting in the DTD schema of permitted data (or in other words, DTD view like in [22], [33]) and hidden rules of *DTD*-to-*DTD view* transformation. This pair (DTD view, hidden rules) can be used to construct a *materialized* version of XML document by deleting forbidden nodes so that the *materialized view* be isomorphic to the *authorized view*. Finally, the *materialized view* is used for evaluation of user queries. We call such a query answering schema *view materialization use case*. It is conceptually represented on the central part of Fig. 1. There are two ways to implement this use case in client-server architecture (where *client* tries to get an access to the XML document and possesses a schema of available data, and *server* stores the initial XML document and must satisfy requests of *client* to this document): server either stores every materialized view or recalculates a corresponding view on the fly every time client issues a query. In the former case, the integrity maintenance becomes unfeasible. In the latter case, the view materialization can be very time- and memory- consuming that is unacceptable in the presence of many users requesting the database simultaneously.

To overcome limitations of the *view materialization* use case, we developed a *query rewriting use case* when *user*'s queries are answered without view materialization: we rewrite a user query formulated for the DTD view into an equivalent query formulated for the original DTD using *hidden rules* [31] and evaluate the resulting query over the original XML database. The schema of such an approach is shown on the right part of Fig. 1.

Hidden rules are defined for every edge of DTD view and describe the path between nodes of that edge in the original DTD. For example, if hidden rule $\sigma(A, B) = B$, node B is a child of node A in the original DTD; if $\sigma(A, B) = C[D]/B$, B is a grandchild of A in the original DTD and the parent of B should be C with a child D.

3 Query Rewriting Algorithm Description

The algorithm for query rewriting has two phases: query parsing and further translation of the parsed query into σ-functions. Query parsing phase implies that user query is represented as a tree of subqueries (*parse tree*) according to the grammar that we have shown in Def. 1. The graphic representation of a generic parse tree is show in Fig. 2. Namely, *basic block* is one of $\langle xpath \rangle$, $\langle path \rangle$, $\langle step \rangle$, $\langle qual \rangle$, or $\langle test \rangle$ which is θ::λ where λ is either an element type name or an asterisk $*$; *optional part* is self-explaining (i.e., $\langle xpath \rangle$ may consist of one or more $\langle path \rangle$s that may consist of one or more $\langle step \rangle$s that may include zero or more $\langle qual \rangle$s); *extended basic block* is related to a qualifier of the form $\langle path \rangle$=*const*; *expression* is related to qualifiers including `and`, `or`, and `not` operators with

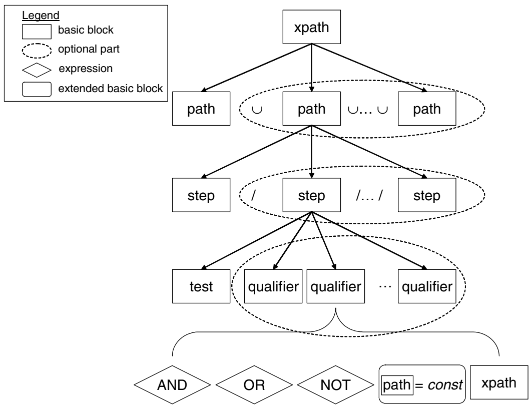


Fig. 2. Generic parse tree

operands as other qualifiers. If $\langle xpath \rangle$ or $\langle path \rangle$ occur in a qualifier, they are parsed as basic blocks according to Def. 1. Leaves of the parse tree are $\langle test \rangle$s.

The translation of the parsed query starts from the leaves of the parse tree and moves up to the root $\langle xpath \rangle$. In particular, for each subquery p and some element A, the algorithm calculates $QR_{(p,A)}$ using Query Rewrite(p_i, B_j), where p_i is a direct subquery (child in a parse tree) of p and B_j is a node reachable from A via p_i in D_v. At the same time, the algorithm calculates $reach(p, A)$ representing the set of nodes reachable from node A via the path p. To obtain a rewriting of the initial user query q, we invoke Query Rewrite(q, `root`).

We do not provide here a formal presentation of Query Rewrite algorithm since it was presented in [31]. Basically, the following rules characterize a process of query rewriting:

1. rewriting of $\langle xpath \rangle$ is a union of rewritten $\langle path \rangle$s that constitute it;
2. rewriting of $\langle path \rangle$ is a rewriting every its step w.r.t. a $\langle test \rangle$ of a previous $\langle step \rangle$. Namely
   - $test_1/$`child`$::test_2 \rightarrow \sigma(test_1, test_2)$;
   - $test_1/$`parent`$::test_2 \rightarrow \sigma^{-1}(test_2, test_1)$;
   - $test_1/$`descendant-or-self`$::test_2 \rightarrow$Query Rewrite($test_1$,$path(test_1, test_2)$), $path(test_1, test_2)$ is a path from $test_1$ to $test_2$ expressed by `child` axes;
   - $test_1/$`ancestor-or-self`$::test_2 \rightarrow$Query Rewrite($test_1$,$path^{-1}(test_1, test_2)$), $path^{-1}(test_1, test_2)$ is a path from $test_1$ to $test_2$ expressed by `parent` axes.

   Here, if $test_2$ is $*$ then $test_2$ is any node v that is in a corresponding relation with $test_1$. The resulting rewritten expression, then, is a union of rewritten expressions as if v where at the position of $test_2 = *$.
3. rewriting of $\langle test \rangle$ with $\langle qual \rangle$ is a rewriting of $\langle qual \rangle$ w.r.t. $\langle test \rangle$;
4. rewriting of `and`/`or`/`not` expressions is respectively `and`/`or`/`not` expression of rewritten subexpressions of the initial expression.

Here $\sigma^{-1}(A, B)$ is, basically, $\sigma(A, B)$ that is inverted with the help of `parent` axis so that to represent a path from B to A (which may be an ancestor in the original DTD) instead of from A to B. For example, if $\sigma(A, B)$=$A/C[D]/B$, user query contains a fragment $B/$`parent`$::A$, then this fragment will be rewritten as $B/$`parent`$::C[D]/parent::A$[Query Rewrite(`root`, $path$(`root`, A))]. Note that `parent`$::A$ is extended with a qualifier that guarantees that A is indeed accessible.

If a set of steps of $\langle path \rangle$ can be represented as an array then we can divide that array in two equal subarrays (one contains the left subpath, another - the right one), we can recursively rewrite subpath corresponding to those subarrays and join them in $O(n)$ time, where n is a number of nodes reachable by the left subpath from `root` node. If the path contains a reverse `parent` axes, the inversion of a corresponding σ-function will take $O(k)$ operations, where k is the number of steps in σ-function. Hence, the overall complexity will be no more than $O((n \times k \times \log m)$, where m is a number of steps in path including those in qualifiers. Since, n and k are limited numbers much smaller that the number of elements in *non-recursive* DTD, the complexity of algorithm may be considered $O(\log m)$.

The closest approach to query rewriting is presented by Fan et al. in [14]. The main differences are:

1. Our algorithm derives a security view without any dummy element types which may be a source of sensitive information leakage. Therefore, the σ-function used in our query rewriting has different semantics.
2. We use an extended XPath fragment has `parent` and `descendant-or-self` axes.
3. Fan et al. use dynamic programming so that $QR_{(q,A)}$ is calculated for *every* DTD element type A; while we perform a rewriting of q w.r.t. to a *subset of relevant element types A of DTD* in a recursive manner. Thus, memory is saved (in [14], memory can be *always* evaluated as $\Omega(m \times |D|)$, where m is a number of all steps in the path including those in qualifiers. In our approach, there are cases when $\Theta(m)$ of memory is required).
4. Moreover, we use divide-and-conquer approach that is faster than simple step-by-step lookup proposed in [14](takes $O(n^2)$ time)
5. However, we do not consider recursive DTDs, while Fan at al. [14] does.

4 Experimental Results

DTD document. In our experimental framework, we used the XMark benchmark [1] that provides the DTD schema `auctions.dtd` which describes an auction scenario. It defines around 75 elements describing a list of auction items, information about bidders, sellers, buyers, etc. The schema of `auctions.dtd` is shown on Fig. 3.

Security Annotation. Fig. 4 shows annotation for every registered user of auctions portal (i.e. user that can participate in auction).

More precisely, the first rule (1) prohibits an access to the information about items and their location in regions. Then, this rule is overridden so that items located in North America are visible for people from United States or Canada (policy 2) and items from United States that can be shipped not only within the country are visible for everyone (policy 3).

Next, information about seller/bidder of any open auction (rules 4 and 5 respectively) is visible if the viewer is seller/bidder himself. In the same way, information on buyer is protected in closed auctions (policy 8). All increases of any particular auction is publicly visible (policy 7).

Rule 9 says that `person` may get a personal information only about himself/herself, i.e. person id should be equal to user login. The latter is expressed by a dynamic variable `$login`. The instantiation of this variable is hardcoded, i.e. if qualifier contains substring `$login`, it should be replaced by the login name passed as a program parameter or taken as a system variable. A personal

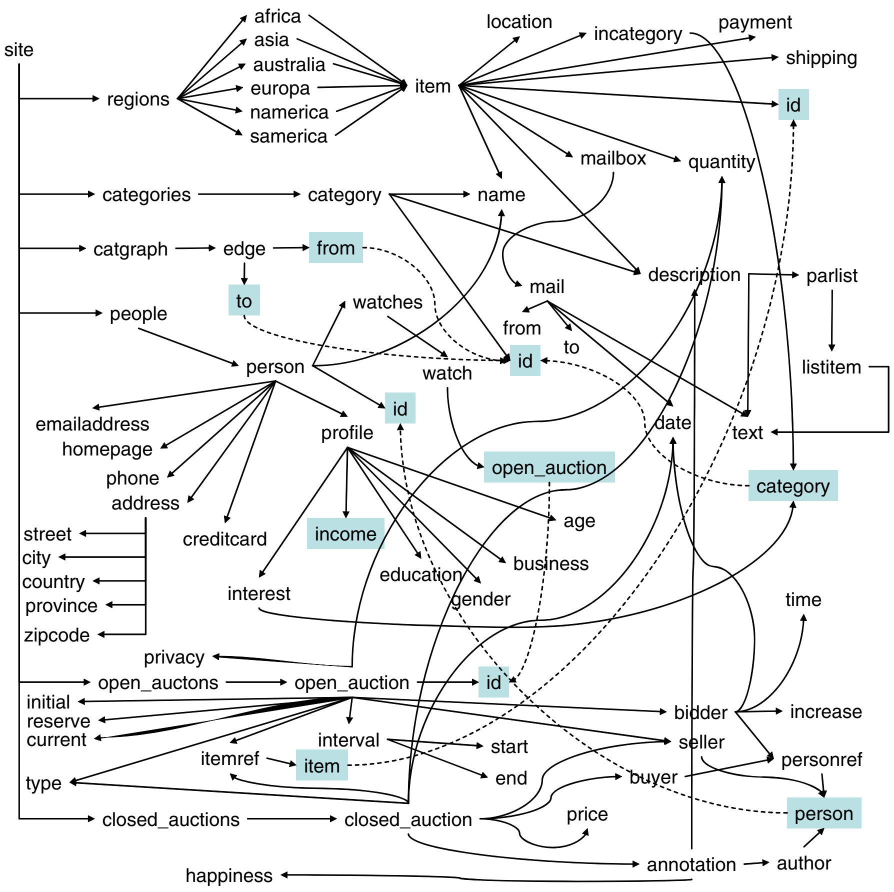


Fig. 3. Schema of the file `auctions.dtd`

profile, instead, can be publicly available if field `business` has a text value *Yes*. However, `business` is not visible in its turn.

Queries. We have defined a set of queries (see Fig. 5) to test both the algorithm Query Rewrite and the advantages of query evaluation w.r.t. different use cases (see Fig. 1).

1 :	$\mathsf{ann}(site, regions)$	$= \mathsf{N}$
2 :	$\mathsf{ann}(regions, namerica)$	$= \mathsf{Q}[/site/people/person[@id = \$login]/address/country/text() =' UnitedStates'$ `or` $/site/people/person[@id = \$login]/address/country/text() =' Canada']$
3 :	$\mathsf{ann}(namerica, item)$	$= \mathsf{Q}[location/text() =' United\ States'$ `and` $shipping/text()! =' Will\ ship\ only\ within\ country']$
4 :	$\mathsf{ann}(open\_auction, seller)$	$= \mathsf{Q}[@person = \$login]$
5 :	$\mathsf{ann}(open\_auction, bidder)$	$= \mathsf{Q}[personref/@person = \$login]$
6 :	$\mathsf{ann}(bidder, increase)$	$= \mathsf{Y}$
7 :	$\mathsf{ann}(closed\_auction, price)$	$= \mathsf{Q}[$`parent ::` $closed\_auction/(buyer/personref$ `or` $seller)/@person = \$login]$
8 :	$\mathsf{ann}(closed\_auction, buyer)$	$= \mathsf{Q}[personref/@person = \$login]$
9 :	$\mathsf{ann}(people, person)$	$= \mathsf{Q}[@id = \$login]$
10 :	$\mathsf{ann}(person, profile)$	$= \mathsf{Q}[business/text() =' Yes']$
11 :	$\mathsf{ann}(profile, business)$	$= \mathsf{N}$

Fig. 4. Annotation for a registered user of an auctions portal

$q_1 : regions/africa$
$q_2 : namerica/item$
$q_3 : people/person/profile/business$
$q_4 : open\_auctions/open\_auction[initial/text() <' 50' \texttt{ and } current/text() >' 100']/increase$

$q_5 : people/person/*$
$q_6 : open\_auctions/open\_auction/*/*$

$q_7 : people/person/profile[@income > 85000]/\texttt{parent} :: person/name$
$q_8 : people/person/name/\texttt{parent} :: person/address/$
$\texttt{parent} :: person/profile[@income > 85000]/\texttt{parent} :: person/name$

Fig. 5. Set of queries

Query q_1 and q_2 will try to get an access to items in African and North American regions respectively. If we assume that the user is from American region, the first query will return nothing, while the second query will get the information about items in North America. Next, q_3 should always return empty set because of the policy rule 11. "Long" query q_4 looks for increases of auctions with the particular start and current prices. Queries q_5 and q_6 introduce asterisk $*$. Reverse `parent` axis is introduced in q_7 that tries to get a name of a person whose income is more than 85K. The result set of this query evaluation should be the same as of the query q_8 that includes three reverse `parent` axes.

Some rewritten queries are depicted on Fig. 6. More precisely, in q_2' before `namerica`, a missing `region` goes. Field `namerica` itself is extended with a qualifier which is $\sigma(regions, namerica)$. In the same way, after `item`, $\sigma(namerica, item)$ goes. The rewriting of q_4' shows that the user will receive `increase`s that were posed by other users, while his own increase values are located under field `bidder` that is visible for him since its person reference is equal to his login. In q_5', asterisk $*$ is rewritten into the union of visible nodes that are in the relation `child` with `people`. The rewriting of q_6' has the same idea. In q_7, we would like to note that `profile` is extended with a security qualifier $\sigma(person, profile)$ in addition to a user-defined filter on it. The second line of q_7' represents reverse axis rewriting. This part is inserted in q_8' three times. Finally, q_1 and q_3 are rewritten to `null` because the user cannot see either `africa` (policy rule 1) or `business` (policy rule 11).

XML documents. XMark data generator [1] produces XML documents conforming to a DTD `auctions.dtd` depicted on Fig. 3. Number and type of elements in resulting XML depend on parameter called *factor*. The significant feature of XMark benchmark is the generation of one unique XML document for one factor value. We generated 20 XML documents with factor $0.0i0$ and $0.0i5$, where $i = \overline{0, 9}$. The size of these XML files varies from 0.3Mb to 9Mb (all together around 100Mb).

Performance results. We tested our implementation on Windows XP platform, Intel Pentium M 1.4GHz, 256Mb DDR SDRAM. For each XML document (recall, we have 20 XML documents), we run evaluation of each query $q_j, j = \overline{1, 8}$ from the viewpoint of 10 users ($login = person_i$, where $i = \overline{0, 9}$) measuring time

q'_2 : $regions/namerica[/site/people/person[@id = \$login]/address/country/text() =' United\ States'$
`or`$/site/people/person[@id = \$login]/address/country/text() =' Canada']/$
$item[location/text() =' United\ States'$ `and` $shipping/text()! =' Will\ ship\ only\ within\ country']$
q'_4 : $open\_auctions/open\_auction[initial/text() <' 50'$ `and` $current/text() >' 100']$
$/bidder[$`not`$(personref/@person = \$login)]/increase$
q'_5 : $people/person[@id = \$login]/(address|watches|phone|name|creditcard$
$|emailaddress|profile[business/text() =' Yes']|homepage)$
q'_7 : $people/person[@id = \$login]/profile[business/text() =' Yes'][@income > 85000]/$
`self` $:: profile[business/text() =' Yes']/$`parent` $:: person[/site/people/person[@id = \$login]]/name$

Fig. 6. Set of rewritten queries

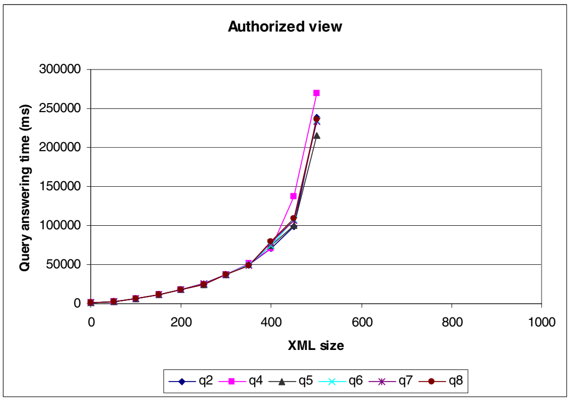
Table 1. Query rewriting performance results

	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8
$person_0$	70	71	60	71	60	60	60	70
$person_1$	60	60	60	60	60	60	70	71
$person_2$	60	70	60	60	60	70	70	70
$person_3$	60	60	60	70	60	70	80	70
$person_4$	60	60	70	70	60	60	70	80
$person_5$	60	60	70	60	70	70	70	80
$person_6$	60	60	70	60	70	70	80	61
$person_7$	70	60	60	80	80	70	80	70
$person_8$	60	60	70	70	70	70	60	70
$person_9$	70	81	60	70	60	60	61	80
avg	63	64.2	64	61.1	65	66	69	72.2

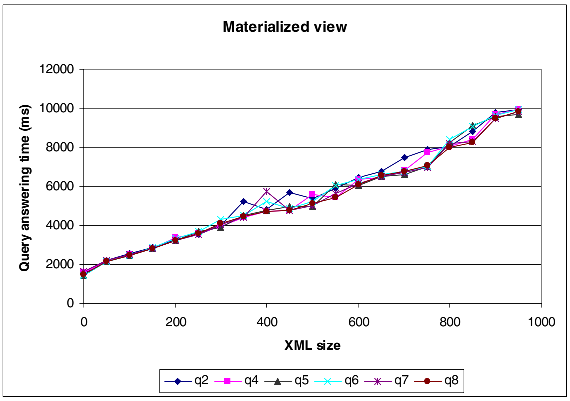
of query rewriting (that includes time of DTD view and σ-function construction) $t^1_{i,j}$ and time of query answering $t^2_{i,j}$.

Query rewriting performance results are shown in Table 1 where each cell (i, j) contains time $t^1_{i,j}$ (in milliseconds) required to rewrite $q_j, j = \overline{1, 8}$ for $person_i, i = \overline{1, 10}$. In all experiments DTD view construction time was between 520 and 620 milliseconds. In the query rewriting part, the measured time includes a query parsing and Query Rewrite running. We used SiXPath [1] processor to parse XPath queries into their tree representation. The processor represents steps of a path in array, so we can apply divide-and-conquer technique that improves efficiency of query rewriting. From Table 3, it can be easily seen that more complicated query requires more time to be rewritten. However, one point is that in reality, the user does not formulate such cumbersome queries like q_8 that, indeed, require "much" time for rewriting. On the other hand, even if the user issues a query that contains time-consuming elements like $*$ or `parent` axes, the query rewriting performance degrades insignificantly (less than 10 milliseconds between the easiest (consisting of 2 location steps) and the hardest (consisting of 10 location steps and of 3 `parent` axes) query in our test case).

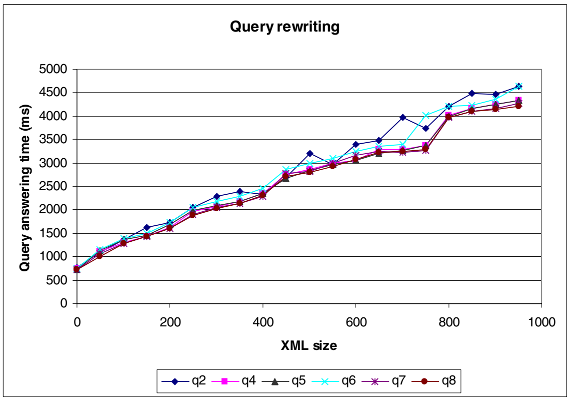
[1] http://sourceforge.net/projects/sixpath



(a) Authorization view use case



(b) Materialized view use case



(c) Query rewriting use case

Fig. 7. Experiments

Query answering time was evaluated by a Xalan [2] XPath evaluator. Unfortunately, XML Task Force evaluator [3] which is shown in [18] as the most efficient and scalable XPath evaluator, could not be used in our test framework because it does not accepts queries that contain union, e.g., like queries q'_5, q'_6. The overall result computation performance is shown on Fig. 7: for each query $q_j, j = \overline{1,8}$, login $person_i, \overline{1,10}$ and each of 20 XML files we performed query evaluation and calculated its average meaning in i, i.e. every dot in flow-chart polylines represents average time for query j and a corresponding XML document. Namely, the Fig. 7(a) shows our experience with XML Access Control Processor (XMLACP) developed by Damiani et al. [13]. Query evaluation includes a preliminary construction of an authorized view which is extremely time consuming. This is because security annotations are propagated on the XML tree. Moreover, security annotation is, basically, a pair $\langle xp, lbl \rangle$, where xp is an XPath expression defining a node being labeled, lbl is Y or N. Since xp may include qualifiers, the step of initial labeling takes much time which grows exponentially with the growth of the initial XML document size. We ceased the conduction of experiments after 11th XML document (500Kb) because of their obvious unfeasibility in terms of time for the larger XML documents.

We have to note that XMLACP approach does not delete forbidden XML nodes if they have permitted descendants, it rather clears their attributes. Therefore, user queries should include even forbidden steps that may reveal sensitive information. We constructed such queries from the output of Query Rewrite but without qualifiers included by a corresponding σ-expression. Another issue is related to a set of authorized views. Their number can be enormous (e.g., we can derive 126 personal views from an XML document computed with factor 0.050, i.e. of the size around 500Kb; this number grows exponentially with increasing XML size on 500Kb) but their sizes is about 10% less than the initial XML document. Therefore, storing all individual views is not feasible even if do not consider integrity maintenance problem.

Materialization use case shown in Fig. 7(b) shows a much better result in time performance. On the other hand, space required for view storing is also reduced since we delete all intermediate N-labeled nodes.

Finally, query rewriting use case is shown in Fig. 7(c). We have some doubts on the affectiveness of query rewriting use case, because a rewritten query may include complicated qualifiers that, in their turn, may require a time-consuming evaluation over the initial XML tree. However, our experiments show that this use case is the best all considered in this paper: it is at least two times faster than the previous case and does not require an additional storage for a materialized view.

Unfortunately, we could not retrieve other existing XML securing systems, e.g., Author-$\mathcal{X}$ [5] (it is lost and cannot be recuperated because of the storing

[2] http://xml.apache.org/xalan-j/

[3] http://www.dbai.tuwien.ac.at/research/xmltaskforce/

computer crash [4].), Lock-X [9] (for the same reason [5].), SMOQE [15] (because of experiment conduction for additional research [6].).

5 Related Work

5.1 Runtime Policy Evaluation

The general scenario of the current category of proposals is the following: The systems defines a set of access control rules of the form $\langle subject, object, action, sign\rangle$ where *subject* is self-explaining, *object* is an XML element/attribute expressed by XPath, *action* is typically `read/write`, $sign \in \{+, -\}$. Different conflict resolution rules and default policy are established as well. With respect to user's request $\langle req\_subject, req\_object, req\_action\rangle$, access control rules applicable to *req_subject* is selected, their *sign*s for *req_action* are propagated to *req_object*. Hence, permission is granted to the user if the resulting sign on *req_object* is +; otherwise, the access is denied.

In particular, [21] introduces the notion of *provisional authorization*, when some provisional action (e.g., logging, transcoding) is performed according to the user's request. The proposal of [7] considers the case when the access control is moved to clients, e.g, secure tokens and smart cards that are used as trust components in different mobile devices (e.g., PC, PDA, cellular phone) participating in applications dealing with sensitive information (e.g., certification, electronic voting, e-payment, health care, etc.). Several papers consider the case of evolving access control policies expressed in XQuery [17] and by means of RDF [2], [19]. Such policies can be used for a derivation of new access control rules including content-based constraints of requested and other documents, environmental information like time and place of request initiator, information about possessed privileges.

Run-time policy evaluation can be accelerated by efficient lookup of compressed accessibility map where compression is performed on objects [37], on objects and actions [20], on objects, subjects and actions [38]. Another direction for improving runtime policy evaluation concerns statical analysis of user queries [28], integration of policy into user query [25], matching user query against efficient policy representation as a tree [29]. In the case when mandatory access control is considered, recursive checks can be reduced by adding special predicates to node tests in the user query [10].

An association between XML nodes can be hidden either at the stage of policy definition [19] or after detecting the possibility of information leakage in security view [36]. Finally, access control for XML documents can be strengthen with a role-based concepts [34].

[4] Personal communication with Elena Ferrari

[5] Personal communication with SungRan Cho

[6] Personal communication with Wenfei Fan

5.2 Security Views for XML

This scenario is called *authorization view use case* in the current paper. The details can be seen, e.g., in [5], [16], [13]. The variation of this use case enforces access control policies cryptographically.

For example, the approach in [6] is based on Author-$\mathcal{X}$ [5]. It avoids generation of multiple physical views for each use by means of different keys for encrypting different portions of the same document. One and only one key is responsible for encryption of each portion of the source XML document. To minimize the number of encryption keys, the portions of the document protected with the same set of policies are encrypted with the same keys. The consequent scenario is key distribution and periodical broadcast of the encrypted document.

Miklau and Suciu extend the Bertino's idea of secure and selective dissemination of XML documents with the notion of *conditional access control rules* [26], which generalizes the term "subject", i.e. authorization is based not on network identifier or user name, but on knowledge presented by the user.

The ideas of [6] and [26] are refined and extended with RBAC in [12].

5.3 Schema-Based Security Views

Stoica and Farkas [33] proposed to produce single-level views of XML when conforming DTD is annotated by labels of different confidentiality levels. The key idea lies in analyzing semantic correlation between element types, modification of initial structure of DTD and using cover stories. Altered DTD then undergoes "filtering" when only element types of the confidentiality lever no higher that the requester's one are extracted. However, the proposal requires expert's analysis of semantic meaning of production rules, and this can be unacceptable if database contains a large amount of schemas which are changed occasionally. No query rewriting is discussed.

Another view-based approach is proposed by Fan et al in [14]. In [22], we refined this solution with other DTD view derivation and XML view materialization algorithms. In this paper, we underlined the similarities and the differences with our approach to query rewriting.

The recent approach to schema-based security views was presented in [27]. The solution allows a complete restructuring of a DTD and relies on a command-like specification language. However, it was mentioned in [27] that many operations are not commutative and have restrictions that means a possibility of errors while designing access control policies. A sophisticated query rewriting is provided.

6 Conclusion

In this paper, we have studied the performance of answering queries on an XML database, subject to access control annotations applied on the original DTD. We show that the query rewriting approach compared to the case of authorized/materialized view is more efficient in sense of time and space.

Time effectiveness takes place because we avoid view materialization which is a time consuming operation. In our experimental benchmark the query rewriting strategy retrieves answer for user query at least 2 times faster than in the case the materialized view and at least 100 faster than in the case of authorized view. Another mentioned point is the space preserving property of advanced method. We recall that the number of views can be extremely large and their size may, in the very good case, be 50% smaller w.r.t. the initial XML document. This may cause problems with the disc space allocation and with the maintenance of data integrity.

Our future research direction is related to drawbacks of the suggested labeling mechanisms outlined below:

1. Hardcoding of user attributes in qualifiers. If at some stage behavior of the system is changed, e.g. from identity based authorization to credential based one (i.e. login may be substituted by private key or social security number, or even by a boolean combination of these attributes), we need to rewrite and recompile the source code responsible for parsing a set of program parameters.
2. The suggested approach of security specification lacks flexibility in defining access rules based on user credentials (i.e. distributed environment with multiple users unknown in advance) and purposes for data storage/access (i.e. privacy issues).
3. Multiple labels of the same semantics but of the different syntax complicate variation of policies.
4. The same syntactically equal qualifiers have the same semantics. Therefore, a change in one qualifier requires revisiting of all the same qualifiers.
5. We are going to investigate the issue of ID/IDREF attributes. Namely, we want to answer the question: what happens if ID attribute should be deleted but its referencing IDREF attribute should exist? Answering this question will help us to extend XML security views to a distributed/fedrated/shared environment.
6. Finally, we are planning to extend our view derivation and query rewriting algorithms for the case of recursive DTDs.

Acknowledgements. This work has been partially supported by EU commission under the SERENITY project. I would also like to thank Gabriel Kuper and Fabio Massaci for their valuable advises regarding query rewriting algorithm.

References

1. XMark – An XML Benchmark Project. `http://monetdb.cwi.nl/xml/index.html`
2. Anutariya, C., Chatvichienchai, S., Iwaihara, M., Wuwongse, V., Kambayashi, Y.: A rule-based XML access control model. In: RuleML, pp. 35–48 (2003)
3. Benedikt, M., Chan, C., Fan, W., Rastogi, R., Zheng, S., Zhou, A.: DTD-directed publishing with attribute translation grammars. In: Bressan, S., Chaudhri, A.B., Lee, M.L., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, Springer, Heidelberg (2003)

4. Bertino, E., Jajodia, S., Samarati, P.: A flexible authorization mechanism for relational data management systems. ACM Transactions on Information Systems (TOIS) 17(2), 101–140 (1999)
5. Bertino, E., Braun, M., Castano, S., Ferrari, E., Mesiti, M.: Author-X: A Java-based system for XML data protection. In: Proceedings of the IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security, pp. 15–26. Kluwer Academic Publishers, B.V (2001)
6. Bertino, E., Ferrari, E.: Secure and selective dissemination of XML documents. ACM Transactions on Information and System Security (TISSEC) 5(3), 290–331 (2002)
7. Bouganim, L., Ngoc, F.D., Pucheral, P.: Client-based access control management for xml documents. In: Proceedings of the 30th Conference on Very Large Data Bases (VLDB'04), pp. 84–95 (2004)
8. Boulahia-Cuppens, N., Cuppens, F., Gabillon, A., Yazdanian, K.: Multiview model for object-oriented database. In: Proceedings of the Annual Computer Security Applications Conference, pp. 222–231 (1993)
9. Cho, S., Amer-Yahia, S., Lakshmanan, L.V.S., Srivastava, D.: LockX: a system for efficiently querying secure XML. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD'03), pp. 669–669. ACM Press, San Diego, California (2003)
10. Cho, S., Amer-Yahia, S., Lakshmanan, L.V.S., Srivastava, D.: Optimizing the secure evaluation of twig queries. In: Bressan, S., Chaudhri, A.B., Lee, M.L., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, pp. 490–501. Springer, Heidelberg (2003)
11. Clark, J., DeRose, S.: XML path language (XPath) version 1.0. w3c recommendation (1999), `http://www.w3.org/TR/xpath`
12. Crampton, J.: Applying hierarchical and role-based access control to XML documents. In: Proceedings of ACM Workshop on Secure Web Services (SWS'04), Fairfax, VA, USA, ACM Press, New York (2004)
13. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for XML documents. ACM Transactions on Information and System Security (TISSEC) 5(2), 169–202 (2002)
14. Fan, W., Chan, C.-Y., Garofalakis, M.: Secure XML querying with security views. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD'04), pp. 587–598. ACM Press, New York (2004)
15. Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: SMOQE: a system for providing secure access to XML. In: SMOQE: a system for providing secure access to XML. VLDB Endowment, pp. 1227–1230 (2006)
16. Gabillon, A., Bruno, E.: Regulating access to XML documents. In: Proceedings of the IFIP TC11/WG11.3 fifteenth annual working conference on Database and application security, Niagara, Ontario, Canada, pp. 299–314. Kluwer Academic Publishers, Dordrecht (2001)
17. Goel, S.K., Clifton, C., Rosenthal, A.: Derived access control specification for XML. In: Proceedings of the 2nd ACM Workshop On XML Security (XMLSEC'03), pp. 1–14. ACM Press, New York (2003)
18. Gottlob, G., Koch, C., Pichler, R.: Efficient algorithm for processing XPath queries. In: Bressan, S., Chaudhri, A.B., Lee, M.L., Yu, J.X., Lacroix, Z. (eds.) CAiSE 2002 and VLDB 2002. LNCS, vol. 2590, Springer, Heidelberg (2003)
19. Gowadia, V., Farkas, C.: RDF metadata for XML access control. In: Proceedings of the 2nd ACM Workshop On XML Security (XMLSEC'03), Fairfax, Virginia, pp. 39–48. ACM Press, New York (2003)

20. Jiang, M., Fu, A.W.-C.: Integration and efficient lookup of compressed XML accessibility maps. IEEE Transactions on Knowledge and Data Engineering (TKDE) 17(7), 939–953 (2005)
21. Kudo, M., Hada, S.: XML document security based on provisional authorization. In: Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS'00), pp. 87–96. ACM Press, New York (2000)
22. Kuper, G., Massacci, F., Rassadko, N.: Generalized XML security views. In: Proceedings of the tenth ACM symposium on Access control models and technologies (SACMAT'05), pp. 77–84. ACM Press, New York (2005)
23. Lunt, T.F., Schell, R.R., Shockley, W.R., Heckman, M., Warren, D.: A near-term design for the SeaView multilevel database system. In: Proceedings of IEEE Symposium on on Security and Privasy (SSP-88), pp. 234–244. IEEE Computer Society Press, Los Alamitos (1988)
24. Lunt, T.F., Denning, D.E., Schell, R.R., Heckman, M., Shockley, W.R.: The SeaView security model. IEEE Transactions on Software Engineering (TOSE) 16(6), 593–607 (1990)
25. Luo, B., Lee, D., Lee, W.-C., Liu, P.: QFilter: Fine-grained run-time XML access control via NFA-based query rewriting. In: Proceedings of the thirteenth ACM international conference on Information and knowledge management (CIKM'04), pp. 543–552. ACM Press, New York (2004)
26. Miklau, G., Suciu, D.: Controlling access to published data using cryptography. In: Aberer, K., Koubarakis, M., Kalogeraki, V. (eds.) Databases, Information Systems, and Peer-to-Peer Computing. LNCS, vol. 2944, pp. 898–909. Springer, Heidelberg (2004)
27. Mohan, S., Sengupta, A., Wu, Y., Klinginsmith, J.: Access control for XML - a dynamic query rewriting approach. In: Proceedings of the 32th Conference on Very Large Data Bases (VLDB'06). VLDB Endowment, Seoul, Korea, pp. 1–12 (2006)
28. Murata, M., Tozawa, A., Kudo, M., Hada, S.: XML access control using static analysis. In: Proceedings of the 10th ACM Conference on Computer and Communication Security (CCS'03), pp. 73–84. ACM Press, New York (2003)
29. Qi, N., Kudo, M.: XML access control with policy matching tree. In: de Capitani di Vimercati, S., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 3–23. Springer, Heidelberg (2005)
30. Qian, X.: View-based access control with high assurance. In: Proceedings of the 1996 IEEE Symposium on Security and Privacy (SP'96), Washington, DC, USA, p. 85. IEEE Computer Society Press, Los Alamitos (1996)
31. Rassadko, N.: Policy classes and query rewriting algorithm for XML security views. In: Damiani, E., Liu, P. (eds.) Data and Applications Security XX. LNCS, vol. 4127, pp. 104–118. Springer, Heidelberg (2006)
32. Stachour, P.D., Thuraisingham, B.: Design of LDV: A multilevel secure relational database management system. IEEE Transactions on Knowledge and Data Engineering (TKDE) 2(2), 190–209 (1990)
33. Stoica, A., Farkas, C.: Secure XML views. In: Proceedings of the 16th International Conference on Data and Applications Security (IFIP'02). IFIP Conference Proceedings, vol. 256, pp. 133–146. Kluwer, Dordrecht (2002)
34. Wang, J., Osborn, S.L.: A role-based approach to access control for XML databases. In: Proceedings of the 9th ACM symposium on Access control models and technologies (SACMAT'04), pp. 70–77. ACM Press, New York (2004)
35. Wilson, J.: Views as the security objects in a multilevel secure relational database management system. In: Proceedings of IEEE Symposium on Security and Privacy (SSP'88), pp. 70–84. IEEE Computer Society Press, Los Alamitos (1988)

36. Yang, X., Li, C.: Secure XML publishing without information leakage in the presence of data inference. In: Proceedings of the 30th Conference on Very Large Data Bases (VLDB'04), pp. 96–107 (2004)
37. Yu, T., Srivastava, D., Lakshmanan, L.V.S., Jagadish, H.V.: A compressed accessibility map for XML. ACM Transactions on Database Systems (TODS) 29(2), 363–402 (2004)
38. Zhang, H., Zhang, N., Salem, K., Zhuo, D.: Compact access control labeling for efficient secure XML query evaluation. In: Proceedings of the 21st International Conference on Data Engineering Workshops (ICDEW'05), p. 1275 (2005)

Answering Queries Based on Imprecision and Uncertainty Trade-Offs in Numeric Databases*

Alexander Brodsky[1,2], Lei Zhang[1], and Sushil Jajodia[1]

[1] Center for Secure Information Systems
George Mason University Fairfax, VA, USA
[2] Dept. of Computer Science
George Mason University Fairfax, VA, USA
{brodsky,lzhang8,jajodia}@gmu.edu

Abstract. Considered in this paper are numeric databases, whose instances are vectors in $\mathcal{R}^n$, queries involve their linear transformations, and imprecise query answers are intervals. Introduced for the first time is a security requirement function ρ for a query, to specify the maximum probability $\rho(l)$ that the precise query answer is within any interval of size l. Developed are random disclosure algorithms that satisfy security requirement functions, and guarantee, under certain conditions, maximum data availability.

Keywords: numeric database; database security; data availability; imprecision; uncertainty.

1 Introduction

Controlling information disclosure is a major task in database security. The main concern in information disclosure is the balance between confidentiality and data availability. This balance, in turn, is based on the fundamental question of how confidentiality and data availability are measured.

There has been extensive work on disclosure control. Many disclosure control methods, including MAC, DAC and RBAC, (e.g., [5,8,14,3]) are based on the notion of information objects of certain granularity (along with other concepts such as subjects or roles), and make binary decisions on whether or not to disclose objects. However, when the system decides not to disclose an object, no partial information is given.

There have been disclosure control methods, including k-anonymity and l-diversity (e.g., [12,13,11]), that use uncertainty, explicitly or implicitly, in answering queries. For example, the user may not know the salary of a person, but know that the salary is one of three possible values. However, uncertainty alone is not enough, when the data is numeric. For example, if the three possible values of salary differ only by a small amount, essentially the salary has been disclosed.

* This material is based upon work supported by National Science Foundation under grants CT-0627493, IIS-0242237, and IIS-0430402; and by Army Research Office under the grant W911NF-07-1-0383.

To deal with the numeric data disclosure, some works (e.g., [4,10,15]) suggested making answers imprecise. For example, instead of answering a person's salary, the system will answer an interval, e.g., from \$35000 to \$65000. However, the imprecise answer may not be good enough, because the user may have additional probabilistic knowledge, such as salary distribution, and, using this distribution, be able to infer that the salary of a person is within a very small interval with relatively high probability.

To remedy the last problem, there has been work (e.g., [9,2,6]) that adopted statistical approach considering both the confidence (for measuring uncertainty) and confidence intervals (to reflect imprecision), in both answering queries and specifying security requirements.

We believe however, that the approach of confidence intervals is not sufficient to express a security requirements that the smaller the confidence interval is, the smaller must be the confidence. For example, if we require the user's confidence of any interval of one person's salary with the size of \$5000 to be equal or smaller than 50%, this requirement does not prevent the user from having a confidence of 50% about a fact that the person's salary is a particular value, says, \$75000. This is precisely the subject of this paper. More specifically, the contributions of this paper are as follows.

We consider a numeric database whose instance consists of n variables ($x_1, \ldots,$ x_n), holding real numbers $v_1, \ldots, v_n$, respectively, i.e., it is a vector $\overline{v}$ in $\mathcal{R}^n$. User queries can request the value of a variable or of a linear arithmetic combination of these variables of the form $c_1x_1 + \ldots + c_nx_n$.

Given a database instance, the system uses a random disclosure algorithm to compute an approximate view $\mathcal{I}$ which is an n-dimensional rectangle in $\mathcal{R}^n$ that contains the database instance $\overline{v}$. We assume that the user knows the random disclosure algorithm used by the system, and the original distribution of $\overline{v}$. Given that, the system also obtains the conditional probability distribution ν of $\overline{v}$ in $\mathcal{I}$, which represents user's implied knowledge. Then, when the user imposes a query q, the system returns the minimal interval I_q, such that the true answer a_q is in I_q with probability 1, according to the user knowledge $(\mathcal{I}, \nu)$.

To the best of our knowledge, this is the first paper to introduce a *security requirement function* $\rho : [0, l_0] \longrightarrow [0, 1]$ for a query q to require that, for any interval of size $l \in [0, l_0]$, the probability that the answer a_q to q is within this interval be bounded by $\rho(l)$, according to the user's implied knowledge ν. We also assume that a fixed set of query views $q_1, \ldots, q_k$ is given, and a security requirement function is given for each query view. Intuitively, query views indicate the relevant or important axes for security and data availability consideration. Note that they can be either from the original axes $x_1, \ldots, x_n$ or their linear combinations, which correspond to other axes in $\mathcal{R}^n$.

We also formally define the notion of *data availability function* by the user-derived knowledge, and the notion of *maximum data availability.* Intuitively, the *maximum data availability* is satisfied by the knowledge $(\mathcal{I}, \nu)$, if it provides *data availability* that is better or same, for every query view, than of any other pair $(\mathcal{I}', \nu')$ that satisfies the security requirement functions for query views.

We first study the basic, one-dimensional case (i.e., one variable x_1, which is the only query view q_1), and assume that the original database instance is uniformly distributed within an interval. For this case, we develop a random disclosure algorithm for which we prove that (1) it satisfies an input security requirement function, and (2) under certain conditions, satisfied with a quantifiable probability, it provides the maximum data availability. We then exemplify how to relax the requirement of uniform distribution of a database instance, but the idea of this extension can be applied to other distributions as well.

Second, we consider the multi-dimensional independent case under the limitation that (1) query views are the original variables $x_1, \ldots, x_n$ (i.e., do not involve linear combinations), and (2) the distributions of dimensions x_i, $i = 1, \ldots, n$ over the projections I_i of $\mathcal{I}$ are independent of each other. For this case, we show how to extend the random disclosure algorithm, so that it will satisfy the same properties, except for a different bound on the probability that maximum data availability is achieved.

Third, we turn to a restricted multi-dimensional dependent in which query views are allowed to be linear combinations. We study in detail a specific setting of two variables x_1 and x_2, three query views x_1, x_2 and $x_1 + x_2$, and the security requirement functions ρ_1, ρ_2 and ρ_3 respectively, defined by $\rho_1(l) = l/l_1$, $\rho_2(l) = l/l_2$, and $\rho_3(l) = l/l_3$, where $l_3 = \lambda_1/\lambda_2(l_1 + l_2)$ and λ_1, λ_2 are positive integers. For this setting, we provide a random disclosure algorithm for which we prove that there exists two-dimensional user knowledge $(\mathcal{I}, \nu)$, that satisfies *maximum data availability* with respect to ρ_1, ρ_2 and ρ_3. We believe that this restricted case is key to understanding the extension to the general dependent case, which is left for future research.

This paper is organized as follows. Section 2 introduces formal definitions of *security requirement function*, user implied knowledge, data availability function, and the notion of maximum data availability, and proves that the user knowledge that provides maximum data availability exists. Section 3 studies the case when dimensions are independent, starting with the one-dimensional case, and Section 4 studies the restricted dependent case. Section 5 concludes and briefly outlines future research directions.

2 Function Based Security Requirement & Data Availability

2.1 A Function-Based Security Requirement

To simplify the discussion, we first consider a one-dimensional numeric database that consists of a single real variable x, which is the only sensitive query view $q = x$. We also assume that the probability distribution of x in interval $[L, R] \subset \mathcal{R}$ is known by users. We denote by v, the current value held by x, i.e, the database instance. In this case, the *random disclosure algorithm* A takes $v \in \mathcal{R}$ as input and randomly produces an interval I that contains v as output, which serves as an approximate view.

Once the result I is returned, the user would have the knowledge that $x \in I$. Also, we make the standard assumption that the random disclosure algorithm is publicly known.

We denote by P_U the user-implied probability function, and by F the user-implied conditional distribution function, i.e., $F(x) = P_U(x \leq v | A(x) = I)$. Let ν be the probability density function (PDF) corresponding to F. Using Bayesian inference, we have:

$$\forall v \in I, \nu(v) = f_{x|A(x)=I}(v) = \frac{f_{I|x=v}(I) f_x(v)}{f_I(I)}$$

where $f_I(I) = \int_L^R f_{I|x=s}(I) f_x(s) ds$ and $f_{I|x=v}(I)$ denotes the density function that A outputs I under the condition that $x = v$.

Thus, we represent the user's knowledge about x as a pair (I, ν) that is provided by A, which we call a *knowledge pair*.

Security requirements of database are to restrict users' knowledge about sensitive values in the database. However, restricting only the size of the interval in user's knowledge pair is not sufficient. For example, suppose that a returned interval is $[0, 1]$, but the user can determine, from the conditional distribution function, that $P_U(v \in [0.999, 1]) = 0.9$. Clearly, situations like this may be unacceptable. One way to avoid such situations is to specify a security requirement as a pair (l, ρ), meaning that:

$$\forall I' \subset \mathcal{R}, sizeof(I') < l \Rightarrow P_U(v \in I') \leq \rho$$

If, for example, $l = 0.1$ and $\rho = 0.5$, then the answer in the previous example would violate this security requirement pair (l, ρ). However, the representation of security requirement as a pair (l, ρ) still may not be sufficiently expressive, because we can not express that the smaller the size of the interval, the smaller the probability should be. For example, given a security requirement pair $(1, 0.5)$, we can not distinguish between the following two situations of user knowledge: (1) x is uniformly distributed in an interval of size 2, and (2) x can be one of two values, having a distance of 1, with 0.5 probability for each.

To be able to differentiate between such situations, we propose to represent a security requirement as a function, that gives probability for each interval length l over a continuous domain.

Definition 1. *A security requirement function for a sensitive variable x is a function $\rho : D \rightarrow [0, 1]$, where the domain D is an interval $[0, l_0]$, such that:*
(1) $\forall l_1, l_2 \in D, l_1 \leq l_2 \Rightarrow \rho(l_1) \leq \rho(l_2)$
(2) $\forall l_1, l_2 \in D, \rho(l_1 + l_2) \leq \rho(l_1) + \rho(l_2)$
(3) $\rho(l_0) = 1$

We say that a knowledge pair (I, ν) satisfies a security requirement function ρ if the following holds:

$$\forall l \in D, a, b \in I, 0 \leq b - a \leq l \Rightarrow P_U(a \leq x \leq b) \leq \rho(l)$$

where $P_U(a \leq x \leq b) = \int_a^b \nu(x) dx$. [1]

[1] Note that, the restrictions that D is an interval and $\rho(l_0) = 1$ can be lifted. These restrictions are made in this paper for the sake of simplicity.

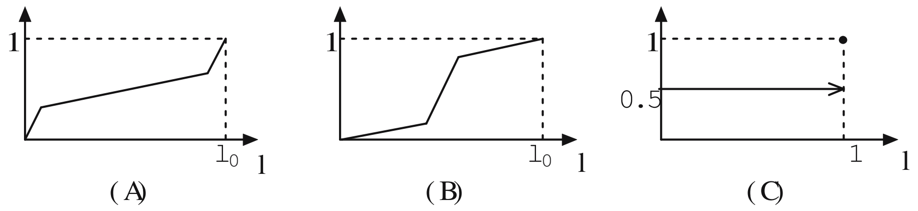


Fig. 1.

It is important to note that any finite set of security requirement pairs can be equivalently represented by a security requirement function. For example, the pair-wise security requirement $(1, 0.5)$ can be represented as a function shown in Figure 1 (C). Note also that Figure 1 (A) shows a possible security requirement function while (B) shows a function that does not satisfy its second property.

2.2 A Function-Based Data Availability Measure

We need to be able to compare different knowledge pairs in terms of *data availability*. More formally,

Definition 2. *A Data Availability Function for a given knowledge pair* (I, ν) *is a function* $\alpha : [0, sizeof(I)] \rightarrow [0, 1]$*, defined by:*

$$\alpha(l) = max_{a,b \in I, b-a=l}\{P_U(a \leq x \leq b)\}$$

where $P_U(a \leq x \leq b) = \int_a^b \nu(x)dx$.

Definition 3. *Given knowledge pairs* (I_1, ν_1) *and* (I_2, ν_2) *and their data availability functions* α_1 *and* α_2*, respectively, we say that* (I_1, ν_1) *provides better data availability than* (I_2, ν_2)*, if:*

– $sizeof(I_1) \leq sizeof(I_2)$*, and*
– $\forall l \in I_1, \alpha_1(l) \geq \alpha_2(l)$

Note that *better data availability* is only a partial order. Yet, we define a strong notion of maximum data availability as follows.

Definition 4. *We say that* (I_1, ν_1) *provides the maximum data availability w.r.t. a security requirement function* ρ *if:*

– (I_1, ν_1) *satisfies* ρ*, and*
– *For any other knowledge pair* (I', ν') *that satisfies* ρ*,* (I_1, ν_1) *provides better data availability than* (I', ν')*.*

The next lemma follows directly from the previous definitions.

Lemma 1. *If the data availability function α of a knowledge pair (I, ν) is identical to a security requirement function ρ, then (I, ν) provides the maximum data availability w.r.t. to ρ.*

We use Lemma 1 to prove the following theorem:

Theorem 1. *If a security requirement function $\rho : D \rightarrow [0, 1]$ is continuous from the right, there exists a knowledge pair (I, ν) that provides the maximum data availability w.r.t. ρ.*

Proof. Because ρ is continuous from the right and the definition of a security requirement function, ρ satisfies the properties of a probability distribution function. By Lemma 1, to prove the theorem, it suffices to construct a knowledge pair (I, ν), such that its data availability function α identical to ρ. We do that as follows: (1)let $I = D$; (2)let ν be the density function of ρ[2]. By definition of data availability function, we have:

$$\forall l \in I, \alpha(l) = max_{a,b \in I, b-a=l}\{\int_a^b \nu(x)dx\}$$

Clearly,

$$\alpha(l) = max_{a,b \in I, b-a=l}\{\int_a^b \nu(x)dx\} \geq \int_0^l \nu(x)dx = \rho(l)$$

On the other hand, by definition of a security requirement function,

$$\forall a, b \in I, b - a = l, \rho(l) \geq \rho(b) - \rho(a) = \int_a^b \nu(x)dx$$

Then,

$$\rho(l) \geq max_{a,b \in I, b-a=l}\{\int_a^b \nu(x)dx\} = \alpha(l)$$

Therefore, $\alpha = \rho$, which completes the proof. □

3 Providing Maximum Data Availability for Independent Cases

3.1 One-Dimensional Case

Consider Algorithm 1, which is a random disclosure algorithm for the one-dimensional case.

Algorithm 1

- Input: (1) the security requirement function ρ which is continuous from the right, with its domain $D = [0, l_0]$; (2) the true value v of x; and (3) an interval $[L, R]$ within which x is uniformly distributed.

[2] Density function of discrete cases can be represented by Dirac Delta Function.

– Output: An interval $I = [v_l, v_r]$.
– Process:
  (1) Construct the knowledge pair (I', ν') using the process in the proof of Theorem 1.
  (2) Randomly select a value v' from $I' = [0, l_0]$ using the PDF ν'.
  (3) Let $v_l = v - v', v_r = v_l + l_0$, output $I = [v_l, v_r]$.

Algorithm 1 assumes that the x is uniformly distributed in an interval $[L, R]$.

Theorem 2. *(1) If $I = [v_l, v_r]$ returned by Algorithm 1 is contained in $[L, R]$, then the corresponding knowledge pair (I, ν) satisfies ρ and, furthermore, provides the maximum data availability w.r.t. ρ; (2) The probability that Algorithm 1 outputs an interval I such that $I \subseteq [L, R]$ is greater than $\frac{R-L-2l_0}{R-L}$.*

Proof. $\forall v \in I$

$$\begin{aligned}\nu(v) &= \frac{f_{I|x=v}(I) f_x(v)}{\displaystyle\int_L^R f_{I|x=s}(I) f_x(s) ds} \\ &= \frac{\nu'(v - v_l) f_x(v)}{f_x(v) \displaystyle\int_L^R f_{I|x=s}(I) ds} \\ &= \frac{\nu'(v - v_l) f_x(v)}{f_x(v) \displaystyle\int_{v_l}^{v_r} v'(s - v_l) ds} \\ &= \frac{\nu'(v - v_l) f_x(v)}{f_x(v)} \\ &= \nu'(v - v_l)\end{aligned}$$

It is easy to see that (I, ν) and (I', ν') have the same data availability function, which, from the proof of Theorem 1, is identical to ρ. Therefore, (I, ν) provides the maximum data availability w.r.t. ρ. Part(2) follows from the fact that if v is within the interval $[L + l_0, R - l_0]$, the output interval will always satisfy $I \subseteq [L, R]$. □

3.2 Special Cases Near the Borders

There is non-zero probability that Algorithm 1 will produce an interval I that is not contained in $[L, R]$. For that reason, the security requirement function ρ may not be satisfied.

To avoid such a situation, one possible amendment to Algorithm 1 is to set an inner interval, $[L', R'] \subseteq [L, R]$. Then, if the output interval $I = [v_l, v_r]$ of Algorithm 1 satisfies the condition $v_l < L'$, then I is replaced with $I_L = [L, L' + l_0]$, similarly, if $v_r > R'$, then I is replaced with $I_R = [R' - l_0, R]$.

Clearly, the user can distinguish whether a "regular" interval I, I_L or I_R is returned. If a regular I is returned, the satisfaction of the security requirement (and maximum data availability) is assured by Theorem 2.

We would also like to guarantee that the security requirement function ρ is satisfied for the cases when I_L or I_R are returned, i.e., that the knowledge pairs

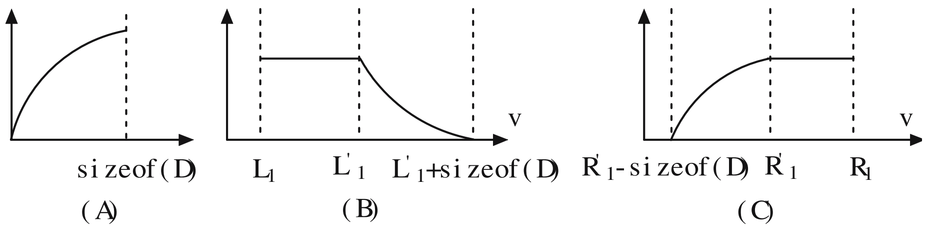


Fig. 2.

(I_L, ν') or (I_R, ν''), respectively, satisfy ρ. For these cases, ν' and ν'' can be computed as follows:

$$\nu'(x) = \begin{cases} \lambda' & L \leq x < L' \\ \lambda'(1 - \rho(x - L')) & L' \leq x \leq L' + l_0 \end{cases}$$

$$\nu''(x) = \begin{cases} \lambda''\rho(x - R' + l_0) & R' - l_0 \leq x \leq R' \\ \lambda'' & R' < x \leq R \end{cases}$$

where λ', λ'' can be determined by solving the following equations:

$$\int_{L}^{L'+l_0} \nu'(x)dx = 1, \quad \int_{R'-l_0}^{R} \nu''(x)dx = 1$$

To illustrate, Figures 2 (B) and (C) show the correlated PDF when the returned answer is either I_L or I_R, respectively, w.r.t. to the security requirement function ρ shown in (A).

To avoid the complexity to compute the least L' and the greatest R' that guarantee satisfaction of ρ by both (I_L, ν') and (I_R, ν''), we provide next common bounds for L' and R' for any possible security requirement function ρ.

Lemma 2. *For any security requirement function ρ:*

(1) if $L' \geq L + 2 \times l_0$, (I_L, ν') satisfies ρ;
(2) if $R' \leq R - 2 \times l_0$, (I_R, ν'') satisfies ρ.

Proof. It is sufficient to show that for any security function $\rho : D \to [0, 1]$ the following holds: $\forall x \in D, \rho(x) \geq \frac{x}{2l_0}$. For $x \in [0, l_0/2]$, it is clear that the inequality above is true. If $\exists x' \in (l_0/2, l_0]$ such that the above inequality is not true, we have $\rho(D) \leq \rho(x)+\rho(D-x) \leq 2\rho(x) < 1$. This contradicts the definition of the security requirement function, which completes the proof. □

This leads to Algorithm 2 given next.

```
Algorithm 2
– Input: (1) the security requirement function $\rho$ which is continuous from right,
  the domain of $\rho$ $D = [0, l_0]$; (2) the true value $v$ of $x$; (3) an interval $[L, R]$
  within which $x$ is uniformly distributed, where $R - L \geq 5l_0$.
– Output: An interval $I = [v_l, v_r]$.
– Process:
  (1) Construct a knowledge pair $(I', \nu')$ using the process in the proof of
      Theorem 1.
  (2) Randomly select a value $v'$ from $I' = [0, l_0]$ with the PDF $\nu'$.
  (3) Let $v_l = v - v', v_r = v_l + l_0$.
  (4) If $v_l < L + 2l_0$, then $v_l = L$ and $v_r = L + 3l_0$.
  (5) If $v_r > R - 2l_0$, then $v_r = R$ and $v_l = R - 3l_0$.
  (4) Output $I = [v_l, v_r]$.
```

Theorem 3. *(1) For any interval $I = [v_l, v_r]$ returned by Algorithm 2, the corresponding knowledge pair (I, ν) satisfies the security requirement function ρ; (2) if $I \subseteq [L+2l_0, R-2l_0]$, then (I, ν) provides the maximum data availability w.r.t. ρ; and (3) The probability that an interval $I = [v_l, v_r]$ returned by Algorithm 2 is contained in $[L + 2l_0, R - 2l_0]$ is greater than $\frac{R-L-6l_0}{R-L}$.*

The proof of Theorem 3 can be easily obtained from Theorem 2 and Lemma 2.

3.3 Pre-Disclosure Algorithm for Different Original Distribution

Until now, we have made the assumption that the database variable x is uniformly distributed on $[L, R]$. One way to relax this assumption and deal with another distribution is to transform it into a new conditional uniform distribution.

Intuitively, the idea is this: the system will use an algorithm, which we call *pre-disclosure algorithm* to disclose some information about x before an approximate query view is computed. The output will be an interval $[L_p, R_p]$ that contains x. The derived knowledge pair will be $([L_p, R_p], \mu_0)$ where μ_0 is a uniform distribution over $[L_p, R_p]$.

To demonstrate the idea, we provide a pre-disclosure algorithm for the following specific setting depicted in Figure 3 A. It shows a possible PDF μ_1 where $\forall 0 \leq x \leq k_0, \mu_1(x) = \frac{x}{k_0}$ and $\forall k_0 \leq x \leq 2k_0, \mu_1(x) = 1 - \frac{(x-k_0)}{k_0}$. For this μ_1, we construct a pre-disclosure Algorithm 3 (which uses Figure 3 (B) in its description):

```
Algorithm 3
– Input: the above PDF $\mu_1$ over $[0, 2k_0]$, $v \in [L, R]$.
– Output: An interval $[L_p, R_p]$.
– Process:
  (1) Construct four line segments $AB, AC, BD, CD$, where:
          $AB : 1 - \mu(x), 0 \leq x \leq k_0$
          $AC : 1, 0 \leq x \leq k_0$
```

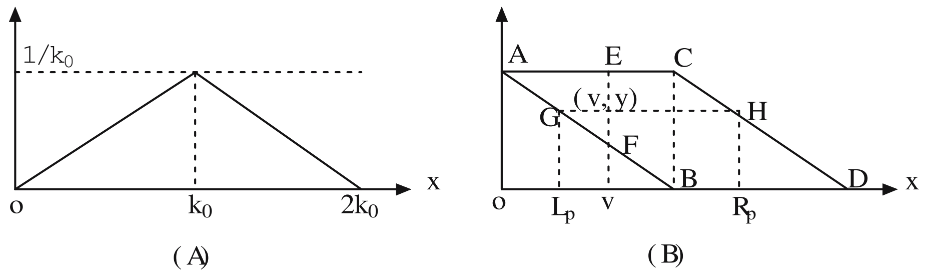


Fig. 3.

$CD : \mu(x), k_0 \leq x \leq 2k_0$
$BD : 0, k_0 \leq x \leq 2k_0$

(2) Let E be the intersection point of AC and $x = v$. Let F be the intersection point of AB and $x = v$.
(3) Random select a point (v, y) from line segment EF with uniform distribution. Draw a line $\mu(x) = y$ that intersects AB at G, intersects CD at H.
(3) Let L_p be the x-coordinate of G and R_p the x-coordinate of H.
(4) Output $I = [L_p, R_p]$.

Theorem 4. *For any interval $I' = [L_p, R_p]$ returned by Algorithm 3, and the corresponding knowledge pair (I', μ'), μ' is the uniform distribution over $[L_p, R_p]$.*

Proof. $\forall v \in I'$

$$\mu'(v) = \frac{\frac{1}{\mu(v)}\mu(v)}{\int_{L_p}^{R_p} \frac{1}{\mu(s)}\mu(s)ds} = \frac{1}{\int_{L_p}^{R_p} 1ds} = \frac{1}{k_0}$$ □

To deal with non-uniform distribution, a pre-disclosure algorithm can be used to produce an output interval $[L_p, R_p]$ within which x is uniformly distributed. Then, Algorithm 2 can be applied using this interval as the input interval $[L, R]$.

Corollary 1. *Let l_{min} be the minimum size of the possible output interval of a pre-disclosure algorithm. Then, for combined application of a pre-disclosure algorithm followed by Algorithm 2, the probability that the resulting knowledge pair will provide maximum data availability w.r.t. ρ is greater than $\frac{l_{min}-6l_0}{l_{min}}$.*

3.4 Independent Multi-dimensional Case

Consider the multi-dimensional case, where a database that consists of n variables $x_1, x_2, \ldots, x_n$. We assume here that the query views $q_1, \ldots, q_n$ are the variables $x_1, x_2, \ldots, x_n$, respectively, and the corresponding security requirement functions are $\rho_1, \ldots, \rho_n$. In this case, a knowledge pair $(\mathcal{I}, \nu)$ is n-dimensional, i.e., $\mathcal{I} \subseteq \mathcal{R}^n$ and ν is a PDF over $\mathcal{I}$. Consider one-dimensional projections $(I_i, \nu_i), i = 1, \ldots, n$, of $(\mathcal{I}, \nu)$, which can be derived as follows:

$$\nu_i(v_i) = \int_{I_n} \ldots \int_{I_1} \nu(v_1, v_2, \ldots, v_n) dv_1 \ldots dv_{i-1} dv_{i+1} \ldots dv_n$$

where $I_1, \ldots, I_n$ are projections of $\mathcal{I}$ on $x_1, \ldots, x_n$. The notion of maximum data availability is a straightforward extension of the one-dimensional case, as follows.

Definition 5. *We say that $(\mathcal{I}, \nu)$ provides the maximum data availability w.r.t. the security requirement functions $\rho_1, \rho_2, \ldots, \rho_n$ if:*

(1) The derived knowledge pair (I_i, ν_i) satisfies ρ_i for any $1 \leq i \leq n$, and
(2) For any other knowledge pair $(\mathcal{I}', \nu')$ that satisfies (1), we have that: for any $1 \leq i \leq n$, the derived knowledge pair (I_i, ν_i) provides better data availability than the derived knowledge pair (I'_i, ν'_i).

If we assume that the query views $x_1, \ldots, x_n$ are independent of each other and uniformly distributed, we can just apply the Algorithm 2 to come with the an interval I_i for each query view, and the corresponding knowledge pair (I_i, ν_i). Clearly, all security requirement functions are satisfied. Consider the n-dimensional knowledge pair $(\mathcal{I}, \nu)$, where $\mathcal{I} = I_1 \times \ldots \times I_n$ and ν is the product of $\nu_1, \ldots, \nu_n$. It is easy to prove that if $\mathcal{I}$ is within the original multi-dimensional rectangle, then it provides maximum data availability w.r.t. $\rho_1, \ldots, \rho_n$. Moreover, the probability of this condition being satisfied is greater than $\prod_{i=1}^{n} \frac{R-L-6l_i}{R-L}$. Similarly, a pre-disclosure algorithms can be used to deal with situations when the original database instance is not uniformly distributed.

4 Dependent Multi-dimensional Case

The situation is more complex when query views can be linear combinations of database variables $x_1, \ldots, x_n$. To understand key ideas of this general case, we consider here a restricted setting where a numeric database consists of two variables x_1, x_2 and query views are x_1, x_2 and $x_3 = x_1 + x_2$. We assume here that x_1 and x_2 are each uniformly distributed over $[L, R] \subset \mathcal{R}$ and mutually independent.

Here the security requirement functions $\rho_1(l) = l/l_1, (0 \leq l \leq l_1)$ for x_1, $\rho_2(l) = l/l_2, (0 \leq l \leq l_2)$ for x_2, and $\rho_3(l) = l/l_3, (0 \leq l \leq l_3)$ for x_3. Next, we show that, under certain conditions, we still are able to provide the maximum data availability for query views x_1, x_2, x_3 w.r.t. their security requirement functions.

Theorem 5. *Let $l_3 = \frac{\lambda_1}{\lambda_2}(l_1 + l_2)$, where $\lambda_1, \lambda_2 \in \mathcal{N}^+$ and $\lambda_1 \leq \lambda_2$. There exists a knowledge pair $(\mathcal{I}, \nu)$, where $\mathcal{I} = [0, L_1] \times [0, L_2]$ in $\mathcal{R}^2$, and ν is a PDF over $\mathcal{I}$, that provides the maximum data availability w.r.t. ρ_1, ρ_2, ρ_3.*

Proof. We prove the theorem by induction, starting with the case when $\lambda_1 = \lambda_2$. As shown in Figure 4 (A), we can construct a knowledge pair $(\mathcal{I}, \nu)$ for (x_1, x_2) such that $\mathcal{I}$ is a 2-dimensional rectangle defined by $[0, l_1]$ and $[0, l_2]$, and ν is a uniform distribution over the diagonal AB. Thus we have three derived knowledge pairs for query views x_1, x_2, x_3 which we call *Basic 3* of the specific l_1, l_2, l_3.

– (I_1, ν_1) for x_1: $I_1 = [0, l_1]$, ν_1 is a uniform distribution over I_1.
– (I_2, ν_2) for x_2: $I_2 = [0, l_2]$, ν_2 is a uniform distribution over I_2.
– (I_3, ν_3) for x_3: $I_3 = [\frac{1}{2}(l_1 + l_2) - \frac{1}{2}l_3, \frac{1}{2}(l_1 + l_2) + \frac{1}{2}l_3]$, ν_3 is a uniform distribution over I_3.

Note that here $l_3 = l_1 + l_2$ implies that $I_3 = [0, l_1 + l_2]$. Clearly, in a *Basic 3*, the data availability function of each (I_i, ν_i) is identical to x_i's security requirement function. Then, by Theorem 1, the claim of Theorem 5 holds for this case.

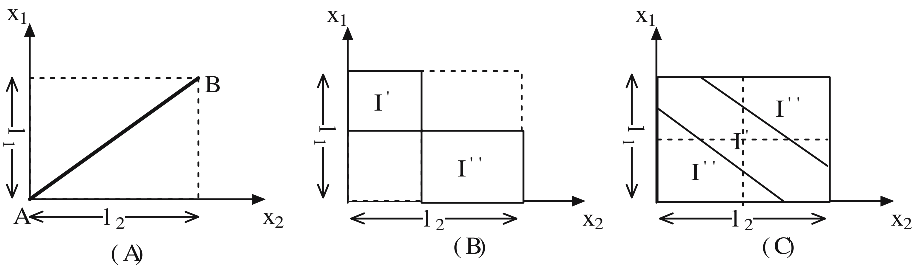


Fig. 4.

Next we prove that: If for any λ_1, λ_2 such that $\lambda_1 < \lambda_2$ and $\lambda_1 + \lambda_2 < k$, there exists a knowledge pair $(\mathcal{I}, \nu)$ that derives a *Basic 3* of $l_1, l_2, l_3 = \frac{\lambda_1}{\lambda_2}(l_1 + l_2)$, then for any λ_1, λ_2 such that $\lambda_1 < \lambda_2$ and $\lambda_1 + \lambda_2 = k$, there exists a knowledge pair $(\mathcal{I}, \nu)$ that derives a *Basic 3* of $l_1, l_2, l_3 = \frac{\lambda_1}{\lambda_2}(l_1 + l_2)$.

– If $\lambda_1 \leq \lfloor \lambda_2/2 \rfloor$, let $\lambda' = \lfloor \lambda_2/2 \rfloor$ and $\lambda'' = \lambda_2 - \lfloor \lambda_2/2 \rfloor$. Then, $\lambda_1 \leq \lambda'$, $\lambda_1 \leq \lambda''$ and $\lambda' + \lambda_1 \leq \lambda'' + \lambda_1 < k$. Thus, based on the assumption, there exist knowledge pairs $(\mathcal{I}', \nu')$ and $(\mathcal{I}'', \nu'')$ such that they derive a *Basic 3* of $l_1, l_2, \frac{\lambda_1}{\lambda'}(l_1 + l_2)$ and a *Basic 3* of $l_1, l_2, \frac{\lambda_1}{\lambda''}(l_1 + l_2)$, respectively.
  Construct a new knowledge pair $(\mathcal{I}, \nu)$ based on "size-reduced" $(\mathcal{I}', \nu')$ and $(\mathcal{I}'', \nu'')$ as shown in Figure 4 (B), as follows.
  • $\mathcal{I}$ is the rectangle defined by $[0, l_1]$ and $[0, l_2]$.
  • $\forall (y_1, y_2) \in \mathcal{I}$:

$$\nu(y_1, y_2) = \begin{cases} \frac{\lambda'}{\lambda_2}\nu'(\frac{\lambda_2}{\lambda'}(y_1 - \frac{\lambda''}{\lambda_2}l_1), \frac{\lambda_2}{\lambda'}y_2) & \frac{\lambda''}{\lambda_2}l_1 \leq y_1 \leq l_1, 0 \leq y_2 \leq \frac{\lambda'}{\lambda_2}l_2 \\ \frac{\lambda''}{\lambda_2}\nu''(\frac{\lambda_2}{\lambda''}y_1, \frac{\lambda_2}{\lambda''}(y_2 - \frac{\lambda'}{\lambda_2}l_1)) & 0 \leq y_1 \leq \frac{\lambda''}{\lambda_2}l_1, \frac{\lambda'}{\lambda_2}l_2 \leq y_2 \leq l_2 \\ 0 & else \end{cases}$$

It is easy to show that $(\mathcal{I}, \nu)$ derives, for x_1, x_2, x_3, a *Basic 3* of $l_1, l_2, l_3 = \frac{\lambda_1}{\lambda_2}(l_1 + l_2)$. [3]

– If $\lfloor \lambda_2/2 \rfloor < \lambda_1 < \lambda_2$, let $\lambda' = \lambda_2 - \lambda_1$ and $\lambda'' = \lambda_2 - \lambda' = 2\lambda_1 - \lambda_2$. Then, $\lambda' + \lambda_2 < k$ and $\lambda'' + \lambda_2 < k$. Thus, based on the assumption, there exist knowledge pairs $(\mathcal{I}', \nu')$ and $(\mathcal{I}'', \nu'')$, such that they derive a *Basic 3* of $l_1, l_2, \frac{\lambda'}{\lambda_2}(l_1 + l_2)$ and a *Basic 3* of $l_1, l_2, \frac{\lambda''}{\lambda_2}(l_1 + l_2)$, respectively.
  Construct a new knowledge pair $(\mathcal{I}, \nu)$ based on "size-reduced" $(\mathcal{I}', \nu')$ and $(\mathcal{I}'', \nu'')$ as shown in Figure 4 (C).
  - $\mathcal{I}$ is the rectangle defined by $[0, l_1]$ and $[0, l_2]$.
  - $\forall (y_1, y_2) \in \mathcal{I}$:

$$\nu(y_1, y_2) = \begin{cases} \frac{\lambda'}{\lambda_2}\nu'(y_1, y_2) & (\frac{1}{2} - \frac{\lambda'}{2\lambda_2})(l_1 + l_2) \le y_1 + y_2 \\ & \le (\frac{1}{2} + \frac{\lambda'}{2\lambda_2})(l_1 + l_2) \\ \frac{\lambda''}{2\lambda_2}\nu''(2y_1, 2y_2) & 0 \le y_1 \le \frac{1}{2}l_1, 0 \le y_2 \le \frac{1}{2}l_2, \\ & (\frac{1}{4} - \frac{\lambda''}{4\lambda_2})(l_1 + l_2) \le y_1 + y_2 \\ & < (\frac{1}{4} + \frac{\lambda''}{4\lambda_2})(l_1 + l_2) \\ \frac{\lambda''}{2\lambda_2}\nu''(2(y_1 - \frac{1}{2}l_1), 2(y_2 - \frac{1}{2}l_2)) & \frac{1}{2}l_1 \le y_1 \le l_1, \frac{1}{2}l_2 \le y_2 \le l_2, \\ & (\frac{3}{4} - \frac{\lambda''}{4\lambda_2})(l_1 + l_2) < y_1 + y_2 \\ & \le (\frac{3}{4} + \frac{\lambda''}{4\lambda_2})(l_1 + l_2) \\ 0 & \textit{else} \end{cases}$$

  It is easy to show that $(\mathcal{I}, \nu)$ derives, for x_1, x_2, x_3, a *Basic 3* of $l_1, l_2, l_3 = \frac{\lambda_1}{\lambda_2}(l_1 + l_2)$. [3]

This completes the inductive proof.

Figure 5 graphically depicts a knowledge pair $(\mathcal{I}, \nu)$ for the case of $l_3 = \frac{2}{3}(l_1 + l_2)$. In it, $\mathcal{I}$ is the rectangle $[0, l_1] \times [0, l_2]$. The PDF ν corresponds to the uniform distribution over the set of points on the bold lines (and is zero elsewhere).

Note that, with the construction method of $(\mathcal{I}, \nu)$ in Theorem.5, we then can construct a similar random disclosure algorithm as Algorithm 2:

Algorithm 4

– Input: (1) The security requirement functions ρ_1, ρ_2, ρ_3 and their domain sizes l_1, l_2, l_3; (2) a database instance (v_1, v_2), (3) a rectangle $[L, R] \times [L, R]$ within which (x_1, x_2) is uniformly distributed.
– Output: Intervals $I_1 = [v_{1l}, v_{1r}]$, $I_2 = [v_{2l}, v_{2r}]$, $I_3 = [v_{3l}, v_{3r}]$ for the three query views.
– Process:
  (1) Construct a knowledge pair $(\mathcal{I}, \nu)$ using the process in the proof of Theorem 5, where $\mathcal{I} = [0, l_1] \times [0, l_2]$.
  (2) Randomly select a point (v_1', v_2') from $\mathcal{I}$ using the PDF ν.

[3] Details can be found in full version of this paper.

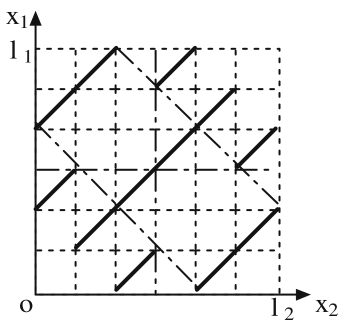


Fig. 5.

(3) Let $v_{1l} = v_1 - v_1', v_{1r} = v_{1l} + l_1$, $v_{2l} = v_2 - v_2', v_{2r} = v_{2l} + l_2$, $v_{3l} = (v_{1l} + v_{1r} + v_{2l} + v_{2r} - l_3)/2, v_{3r} = v_{3l} + l_3$, $signal = 1$.
(4) If $v_{1l} < L + 2l_1$, then $v_{1l} = L$, $v_{1r} = L + 3l_1$, $signal = 0$.
(5) If $v_{1r} > R - 2l_1$, then $v_{1r} = R$, $v_{1l} = R - 3l_1$, $signal = 0$.
(6) If $v_{2l} < L + 2l_2$, then $v_{1l} = L$, $v_{1r} = L + 3l_2$, $signal = 0$.
(7) If $v_{2r} > R - 2l_2$, then $v_{1r} = R$, $v_{1l} = R - 3l_2$, $signal = 0$.
(8) If $signal = 0$, then $v_{3l} = (v_{1l} + v_{2l}), v_{3r} = v_{1r} + v_{2r}$.
(9) Output $I_1 = [v_{1l}, v_{1r}]$, $I_2 = [v_{2l}, v_{2r}]$, $I_3 = [v_{3l}, v_{3r}]$.

Also the following theorem will be straightforward.

Theorem 6. *(1) For any rectangle $\mathcal{I}$ returned by Algorithm 4, the corresponding knowledge pair $(\mathcal{I}, \nu)$ satisfies the security requirement functions ρ_1, ρ_2 and ρ_3; (2) if $\mathcal{I}$ is contained in $[0, l_1] \times [0, l_2]$, then $(\mathcal{I}, \nu)$ provides the maximum data availability w.r.t. ρ_1, ρ_2, ρ_3; and (3) The probability of this condition being satisfied is greater than $\frac{(R-L-6l_1-6l_2)^2}{(R-L)^2}$.*

5 Related Work and Conclusions

Some additional comments on related work are as follows. For the micro-data disclosure problem, the [12,13] were first to provide an implicit measure of uncertainty (k-anonymity). The work [11] introduced the notion of l-diversity, which refined the notion of k-anonymity. The work [15] further refined the notion of uncertainty by defining a measure of distance between sensitive values.

For the information disclosure in numeric databases, [10] defined a measure of imprecision by giving a interval, which can be regarded as a confidence interval with the probability of 1. The work [4] defined a more general notion of imprecision based on volume. The work [9] considered both intervals and conditional probabilities for privacy protection in the context of online auditing.

In data mining, the works [2,6] adopted the idea of confidence intervals to define a privacy breach. The work [1] used information theory measures to quantify disclosed information and define a privacy breach. The work [7] refined this idea by defining privacy-sensitive properties.

To conclude, this paper is the first, to the best of our knowledge, to introduce the notion of security requirement functions, to accurately reflect the imprecision-uncertainty trade-offs. It provided an initial study of random disclosure algorithms that satisfy security requirement functions, and provide, under certain conditions, the maximum data availability. For future work we plan to extend the results to the unrestricted dependent case, as well as to discrete databases. Finally, we would like to thank anonymous reviewers for their useful comments and suggestions.

References

1. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: Proceedings of the 20th Symposium on Principles of Database Systems (2001)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: SIGMOD, pp. 439–450 (2000)
3. Bertino, E., Samarati, P., Jajodia, S.: An extended authorization model for relational databases. IEEE Transactions on Knowledge and Data Engineering 9(1), 85–101 (1997)
4. Brodsky, A., Farkas, C., Wijesekera, D., Wang, X.S.: Constraints, inference channels and secure databases. In: Dechter, R. (ed.) CP 2000. LNCS, vol. 1894, pp. 98–113. Springer, Heidelberg (2000)
5. Bell, D.E., Lapadula, L.J.: Secure computer systems: Mathematical foundations and model. The Mitre Corporation (1975)
6. Dwork, C., Nissim, K.: Privacy-preserving data mining on vertically partitioned databases. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 528–544. Springer, Heidelberg (2004)
7. Evfimievski, A., Srikant, R., Agrawal, R., Gehrke, J.: Privacy preserving mining of association rules. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery in Databases and DataMining, pp. 217–228. ACM Press, New York (2002)
8. Jajodia, S., Samarati, P., Sapino, M.L., Subrahmanian, V.S.: Flexible support for multiple access control policies. ACM Trans. Database Syst. 26(2), 214–260 (2001)
9. Kenthapadi, K., Mishra, N., Nissim, K.: Simulatable auditing. In: PODS, pp. 118–127 (2005)
10. Li, Y., Wang, L., Wang, X., Jajodia, S.: Auditing interval-based inference. In: Proceedings of the 14th International Conference on Advanced Information Systems Engineering, pp. 553–567 (2002)
11. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: ICDE (2006)
12. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. (2001)
13. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, CMU, SRI (1998)
14. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-based access control models. IEEE Computer, 38–47 (1996)
15. Xiao, X., Tao, Y.: Personalized privacy preservation. In: SIGMOD (2006)

Architecture for Data Collection in Database Intrusion Detection Systems

Xin Jin and Sylvia L. Osborn

Dept. of Computer Science,
The University of Western Ontario,
London, Ontario, Canada
{xjin5,sylvia}@springer.com

Abstract. A database intrusion detection system (IDS) is a new database security mechanism to guard data, the most valuable assets of an organization. To provide the intrusion detection module with relevant audit data for further analysis, an effective data collection method is essential. Currently, very little work has been done on the data acquisition mechanisms tailored to the needs of database IDSs. Most researchers use the native database auditing functionality, which excludes privileged users such as database administrators (DBAs) from being monitored. In this paper, we present a new approach to data collection for database IDSs by situating data collecting sensors on the database server and having the data transmitted to the audit server on a physically different site for further processing. This approach can guarantee that behavior of both average users and privileged users are monitored for signs of intrusion.

Keywords: intrusion detection, database IDS, database security.

1 Introduction

Recently, database intrusion detection has attracted a lot of researchers' attention [1,2,3,4,5,6,7,8,9,10,11,12,13,14]. A database-specific intrusion detection system (IDS) can serve as an additional layer of database security, which applies the ideas and results from generic intrusion detection research to detect misuses targeted towards database systems.

An IDS that monitors for intrusive behavior needs to collect data on the dynamic state of the system. These data are then fed to the server for analysis of any signs of intrusion. These security data can take the form of network packets, audit logs of operating systems or SQL statements executed by different users in the database systems scenario.

Currently, there are essentially two approaches to data acquisition in database IDSs. One is to simply use the auditing functionality provided by the database management system (DBMS); the other is to tap into the communications channel between web-based applications and the back-end database server to intercept interesting data.

The former approach naturally comes to mind because the DBMS maintains all the records and statistics related to database performance. The data collection module thus has access to the entire context of the SQL stream, the very type of data most database IDSs inspect for anomalies [1,2,3,4,11,12,13]. However, this approach fails to take into account all the users. It is especially insufficient in exerting control over privileged users like database administrators (DBAs), who, as the ultimate owner of the data, can actually delete and modify the audit logs and even turn off the auditing functionality provided by the DBMS altogether. These privileged users, if corrupt, can potentially cause more damage than average users. In fact, there is an increasing recognition that a significant threat comes from inside, not outside, the organization [15,16,17].

The second approach has only been applied to MySQL, an open-source DBMS, where the library responsible for communication with the database server is modified in order to intercept the interesting SQL statements. For the large majority of DBMSs, especially commercial DBMSs, however, modifying the source is not an option.

In this paper, we propose a new approach for data collection in database IDSs by situating data collecting sensors on the database server and having the data transmitted to the audit server on a physically different site for further processing. This mechanism can exert control over all users without requiring modification to any existing DBMS functionalities.

The rest of the paper is organized as follows. Section 2 briefly introduces database intrusion detection systems. Section 3 describes the three data acquisition methods used by most IDSs. In Section 4, the advantages and disadvantages of each data acquisition method are examined in the database IDS scenario . In Section 5, we present our approach for data collection. Section 6 outlines our future research plans.

2 Database Intrusion Detection

2.1 Intrusion Detection and Types of IDSs

An intrusion is defined as “any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource” [18]. This includes a deliberate unauthorized attempt to access or manipulate the information or render a system unreliable or unusable. Intrusion detection is thus defined as the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusion [19].

Intrusion detection research has been on-going for more than 20 years. However, previous efforts were largely focused on network-based intrusion detection and host-based intrusion detection [1]. Network-based intrusion detection typically works by having one or more sensors on the network monitor network traffic in an attempt to discover if an intruder is trying to break into the system. Host-based intrusion detection, on the other hand, works by monitoring the log files situated in the hosts.

In terms of intrusion detection models, all IDSs are based either on signatures or anomaly models [2]. Anomaly-based intrusion detection works by defining the normal usage behavior patterns to identify intrusions. The normal usage patterns can be constructed from statistical measures of the system features, for instance, the CPU usage and I/O activities by a particular user or an application. Anomaly-based intrusion detection mainly has two modes: a training mode and a detection mode. In the training mode, a normal profile is created by exposing the sensor to non-exploit traffic, while in the detection mode, some distance algorithms are used to compare new data against the normal profile and any distance which crosses some threshold is considered to be an anomaly. Anomaly-based intrusion detection is generally subject to high false positives.

Misuse-based intrusion detection, also known as signature-based intrusion detection, uses well-defined patterns of the attack that exploit weaknesses in systems and applications to identify the intrusions. These explicit patterns, or signatures, are characterized in advance and used to match against the user behavior to detect intrusions. This type of IDSs generally have high false negatives and are not effective against new types of attacks because they can recognize only the ones that have been previously identified and have had signatures written for them.

2.2 Database IDSs

In spite of the significant role of databases in information systems, there is very limited work on intrusion detection in database systems. However, problems posed by intrusions directed to database systems cannot be solved by traditional database security mechanisms or IDSs at the network or host level.

First, as the focus in database intrusion detection is to continuously monitor some dynamic behavior characteristics of the database system to determine if an intrusion has occurred, static security mechanisms such as data encryption, firewalls and virus protection schemes cannot provide this functionality. Neither can this be resolved by traditional database security mechanisms such as authorization, access control, integrity control, etc.

Second, neither network-based nor host-based IDSs can detect malicious behavior from users at the database level, or more generally, the application level, because they do not work at the application layer. Specifically, audits at the system or network level are not suited for intrusion detection at the information level because the semantics of the applications are not reflected in the low-level audit logs [4]. For instance, host-based intrusion detection fails to detect any intrusion directed at databases because users who seek to gain database privileges will likely be invisible at the operating systems level, and thus invisible to the host-based intrusion detectors. Therefore, SQL injection [2] and other SQL-based attacks targeted at databases cannot be effectively detected by network-based or host-based IDSs.

In addition, host- or network-based IDSs are mainly focused on detecting external intrusions as opposed to internal intruders, while, as mentioned earlier, legitimate users who abuse their privileges are the primary security threat in database systems.

Therefore, intrusion detection techniques and models specially designed for databases are becoming an imperative need [1,8,12,14,7]. Ideally, the database-specific IDS should work as a complementary mechanism to the existing network-based and host-based intrusion detection systems rather than replacing them.

3 Related Work

Although some research has been done with respect to the data collection mechanisms of generic IDSs [20,21,22], little work has been done focusing on the data acquisition methods tailored to the needs of database IDSs. This research is important because incomplete, incorrect data or data with a significant delay could lead to erroneous results from the IDS, either generating unbearably high false alarms or giving its users a false sense of security.

3.1 Data Collection Methods for Generic IDSs

Three different types of data acquisition methods for IDSs are mentioned in [20], namely Interception, Instrumentation of a Third Party, and Instrumented Application.

Intrusion detection systems which monitor applications by interception are by far the most common and have the advantage of being easy to deploy. Most network-based intrusion detection systems make use of this method. Existing systems do not need to be modified or reconfigured, and since their monitoring is passive, these IDSs are unobtrusive. However, encrypted traffic may be a problem [21].

Instrumentation of a third party, or the host-based approach, has the obvious advantage of being able to access unencrypted data and system resources such as operating system audit trails and application logs. However, a number of monitors need to be installed instead of just one, thus incurring more administrative overhead. Also, the user could experience a performance penalty as the monitor is on the same host as the application [20]. Furthermore, most monitors at the OS level cannot detect attacks directed at the lower network protocol levels because network information typically does not become available in the audit event stream until it has reached the higher protocol levels.

Instrumented Application, or collecting data directly from the application that is being monitored, also has the advantage of having access to unencrypted information. In addition, this method can take advantage of the application's domain knowledge to select the most relevant data to be collected. However, the application is required to provide an interface to successfully integrate a monitor into the application, otherwise modifications to the application itself may be required. As a result, the development efforts could be significant. This method may also impact the performance when the monitor module is running on the same host as the monitored service.

3.2 Existing Database IDSs and Their Data Collection Methods

Interception with Modification to the DBMS. Valeur et al. present an anomaly-based IDS [11] for the detection of attacks exploiting vulnerabilities in web-based applications to compromise a back-end database. The IDS taps into the communication channel between web-based applications and the database server. SQL queries performed by the applications are intercepted and sent to the IDS for analysis.

This data collection method might look like Interception described in section 3.1, but it relies on modifying the library of MySQL (an open-source database) that is responsible for the communications between the two parties. Thus, this approach cannot be generalized for commercial databases such as MS SQL Server, Oracle, etc.

Built-in DBMS Auditing. Most database IDSs in the literature use the native auditing functionality provided by the DBMS to collect audit data. This includes using built-in auditing tools of the DBMS, manipulating the database log files and invoking DBMS-specific utilities. This method is, in fact, a special case of Instrumenting Applications discussed in section 3.1.

DEMIDS, an anomaly database IDS, is presented by Chung et al. in [5]. It assumes that the access patterns of users typically form some working scopes and it takes into account the domain knowledge about the data structures and semantics encoded in a database schema. The audit data of users are collected by monitoring their queries through the auditing functionality of the DBMS. Spalka et al. [13] adopts a hybrid approach to intrusion detection in relational databases. The IDS consists of three components: an anomaly detector for the database, an anomaly detector for the user interaction, and a misuse detector for the database scheme. User interaction data are collected with the auditing tools of the DBMS. Bertino et al. discuss an anomaly detector for RBAC-administered databases in [12]. The proposed approach is based on mining database traces stored in log files and it couples the IDS with Role Based Access Control (RBAC) [23,24] to detect role-intruders. The IDS uses the database log files to form role profiles of different granularities to identify user behavior.

Utilities such as SQL trace provided by many database vendors are used by several researchers in their database intrusion detection mechanisms to collect and view database events. DIDAFIT, a misuse-based database IDS proposed in [1], detects anomalous database accesses by characterizing legitimate accesses through fingerprinting executed SQL statements using regular expressions. Ramasubramanian and Kannan discuss in [3] a hybrid database intrusion prevention system using a combination of both statistical anomaly prevention and rule based misuse prevention. Both of these IDSs use the sql_trace utility provided by Oracle to capture operations in a database session of a user.

4 Advantages and Disadvantages of Built-in DBMS Auditing

Since the large majority of existing database IDSs use in some degree the built-in DBMS auditing functionality, it is worth investigating the advantages and disadvantages of this method.

4.1 Advantages

Easy to Deploy. Using the "out-of-the-box" auditing functionalities of the DBMS does not involve much development effort in terms of data collection. The sql_trace functionality, for instance, is supported by many commercial databases such as MS SQL Server and Oracle. It is an interface made available through extended stored procedures. By turning on this utility, events can be collected and viewed and even channeled to a designated file.

Accessible to more Comprehensive Information. Native auditing functionality can be extremely powerful. Oracle, for instance, provides exhaustive native audit functionality capable of monitoring and logging practically any database activity.

4.2 Disadvantages

Impact on Performance. The sql_trace utility, for instance, is designed to assess the efficiency of the SQL statements an application runs and optimizing database performance. It is not meant to be used on an on-going basis, especially on a production system, because when enabled it can consume substantial memory, CPU cycles as well as disk space. It is even worse when the native auditing functionality at the database level is enabled because it creates a bottleneck that significantly impacts database performance.

No Intelligence in Certain Native Auditing. Sql_trace utility can either be enabled or disabled. There is no built-in intelligence for data collection and this utility generates a large amount of unwanted audit data for the intrusion detection module.

Complex and Hard to Meet Individual Corporate Auditing Requirements. Oracle Database Server, an auditing utility at the database level, is usually disabled because it takes a significant amount of configuration effort and this must be done individually for each database server. Even when the native auditing functionality is enabled, the logged audit events are rarely reviewed because the data are too voluminous to be useful.

Control of the Database Implies Full Control of the Auditing Functionality. There is no way to stop database administrators from deleting or truncating the audit trail, preventing them from changing the auditing configuration or from disabling the auditing utility altogether. Therefore, no separation of duties is provided, which is the key to the confidentiality, integrity and assurance of any computer system.

5 Proposed Approach to Data Collection in Database IDS

To solve the problems discussed above, we propose a novel approach to data collection in database IDSs.

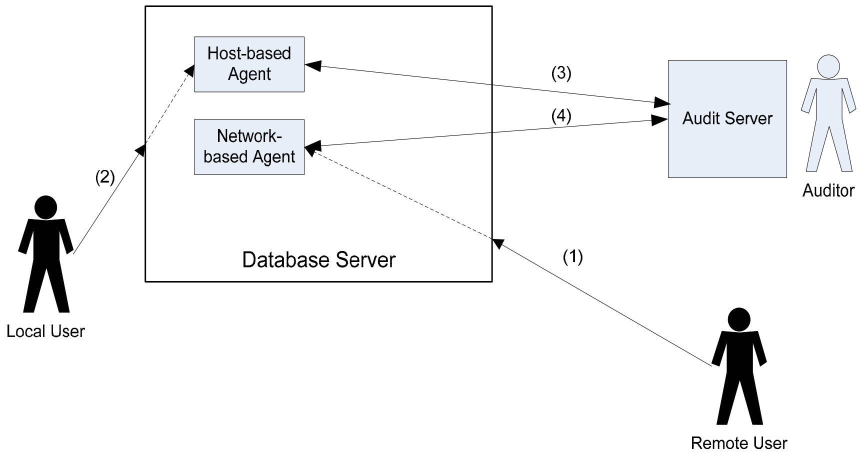


Fig. 1. Architecture of the Proposed Method

5.1 Architecture

The audit data we are interested in collecting are requests submitted by different users to the database server. Here, we distinguish two types of users: local users and remote users. Typical local users are DBAs performing administrative duties on the same machine where the database server resides. Examples of remote users include average database users interacting with the database through applications or utilities, as well as DBAs performing remote database administration through a network connection. Since requests submitted by different types of users go through different channels, separate agents dealing with data collection for each channel are required.

The proposed architecture consists of three major components: the host-based agent, the network-based agent, and the audit server (Fig 1). The network-based agent is responsible for collecting requests submitted by remote users through the network and the host-based agent deals with requests that do not go through the network channel. The audit server is the locus where raw audit data are processed, attempts of intrusions are detected and appropriate actions are taken.

The flows of interaction are as follows:

(1) Requests issued by remote users are captured by the network-based agent;

(2) Requests issued by local users are captured by the host-based agent;

(3) Raw audit data (in the form of users' requests) collected by the host-based agent are sent to the audit server; the audit server checks if the host-based agent is still up and running;

(4) Raw audit data collected by the network-based agent are sent to the audit server; the audit server checks if the network-based agent is still up and running;

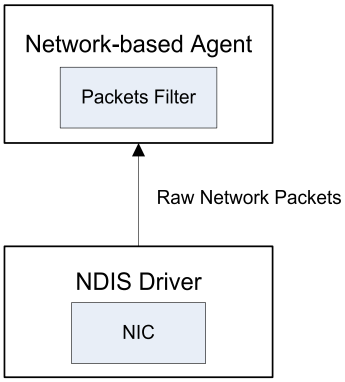


Fig. 2. Working of the Network-based Agent

5.2 Initial Implementation

We have experimented with the proposed architecture using Oracle 10.2.0.1 with more implementation details to follow. Although the current implementation is still primitive, the proposed approach has been proved to work.

The network-based agent and the host-based agent are both installed on the database server and run as independent services on Windows XP. In this way, they require system administration instead of database administration, providing a separation of duties of a system administrator from a DBA.

The network-based agent captures requests submitted by remote users through the network to the database server and then forwards these data to the audit server. The network-based agent works as follows (Fig 2). First, raw data are received by the Network Interface Card (NIC) on the database server. Next, raw network packets are acquired by calling the functions provided by the Network Driver Interface Specification (NDIS) Driver. Interesting data are then obtained by filtering out irrelevant data packets such as those having different protocol encapsulations.

The host-based agent captures requests that do not go through the network. For instance, in Oracle, this can be achieved by reading the executed SQL statements from the System Global Area [25]. The host-based agent requires DBA privilege so that procedures and functions associated with querying the SQL area in the SGA can be successfully executed. Although querying the SQL area can be resource intensive and could affect the performance on a critical production system, this kind of monitoring is only done on local users who are most likely the small handful of DBAs within the organization. Therefore, impact on performance is significantly mitigated.

The audit server checks if the network-based agent and the host-based agent are still up and running by sending a message requiring confirmation from time to time. If either of them is disabled or removed from the database server for some reason, the corresponding response will not be received. As a result, an alarm will be triggered to alert the auditor.

To prevent intruders from hijacking or modifying the audit data transmitted to the audit server, communications channels between the network-based (host-based) agent and the audit server can be secured by encrypting all the traffic between them.

5.3 Advantages

Consistent Control over all Users. Requests from local users can be collected by the network-based agent and those from remote users can be collected by the host-based agent. More importantly, activities of privileged users such as DBAs can now be monitored along with those of average users. Thus, risk of insider threat is considerably reduced.

Minimum Impact on Database Performance. Compared to using native auditing functionality of the DBMS for data collection, the proposed method causes only minimum performance overhead. Data collected at the database server are sent to the audit server after a brief data preprocessing, which filters out irrelevant data and transforms the raw data into formats understandable by the auditing module. All the data analysis and intrusion detection work are carried out on the dedicated audit server. As a result, little performance impact on the database server is incurred, which is especially important in a production environment. The audit server implemented on a separate server can even allow real time analysis without impacting the source database.

Separation of Duties. This data collection module provides a way of separation of duties because the network-based agent and the host-based agent installed on the database server work as services running on the host machine. The system administrator is responsible for keeping these services up and running. DBAs have no control over these services. In this way, system administration instead of database administration is required, enforcing a separation of duties.

Non-obtrusiveness and Adaptiveness. Both the network-based agent and the host-based agent work by passively intercepting data without interrupting the normal run of the database system or enabling any native auditing functionality of the DBMS. In addition, the proposed method does not require any modification to the DBMS or the applications communicating with the DBMS. Thus, it is relatively easy to adapt the data collection module to accommodate various DBMSs without significant development effort.

Minimum Administration. The network-based agent and the host-based agent responsible for data collection are only installed on the database server rather than on each and every host running the DBMS as in traditional host-based data collection. As a result, only negligible administrative work is involved with the proposed approach.

5.4 Discussion

Different data collection approaches for database IDSs arise in part from the discrepancy among researchers on who should be responsible for detecting intrusions targeted at databases.

Some assume that DBAs should be monitoring suspicious behavior aside from performing their usual tasks such as tuning the performance of the DBMS [7]. Then one naturally asks: who should monitor DBAs, who have already been entrusted with an organization's most critical information?

Some discuss their approaches to database intrusion detection without providing an answer to this question. However, their data collection modules have substantially reduced the likelihood that DBAs will be monitored as effectively as average users [12,13,1].

Still some assume a security officer or a security administrator other than the DBA better fits the job [5,26,4]. Even though their data collection modules are insufficient in sustaining this claim, we believe this is a fair assumption because it conforms to the concept of separation of duties, which is the key to the confidentiality, integrity and assurance of any computer system.

6 Conclusion and Future Work

Database IDS is a relatively new topic and research done in this area is still very limited. As in constructing a generic IDS, a data collection mechanism is the first step. In this paper we discuss the data acquisition methods for generic IDS and specifically how the data are collected in the database IDS scenario in various research presented in the literature of database intrusion detection. The advantages and drawbacks of current methods are discussed.

Finally, we present a novel framework for data collection tailored to the needs of database IDSs to solve some of the difficulties encountered by previous research in the field. A primitive version of the architecture has been implemented for Oracle10.2.0.1.

As part of future work, we would like to implement a more sophisticated version of the framework and do more benchmarking on its impact on the performance of the database. Another direction for future research is to design a specific database IDS that takes into account the features of the audit data collected using the proposed approach. We are especially interested in applying data mining techniques to the IDS to detect anomalous database activities from both average users and privileged users.

References

1. Low, W.L., Lee, J., Teoh, P.: DIDAFIT: Detecting intrusions in databases through fingerprinting transactions. In: ICEIS, pp. 121–128 (2002)
2. Rietta, F.S.: Application layer intrusion detection for SQL injection. In: Menezes, R. (ed.) ACM Southeast Regional Conference, pp. 531–536. ACM Press, New York (2006)

3. Ramasubramanian, P., Kannan, A.: Intelligent multi-agent based database hybrid intrusion prevention system. In: Benczúr, A.A., Demetrovics, J., Gottlob, G. (eds.) ADBIS 2004. LNCS, vol. 3255, pp. 393–408. Springer, Heidelberg (2004)
4. Ramasubramanian, P., Kannan, A.: A genetic-algorithm based neural network short-term forecasting framework for database intrusion prediction system. Soft Comput. 10(8), 699–714 (2006)
5. Chung, C.Y., Gertz, M., Levitt, K.N.: DEMIDS: A misuse detection system for database systems. In: IICIS, pp. 159–178 (1999)
6. Lee, S.Y., Low, W.L., Wong, Y.: Learning fingerprints for a database intrusion detection system. In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, Springer, Heidelberg (2002)
7. Lee, V., Stankovic, J., Son, S.: Intrusion detection in real-time database systems via time signatures. In: Proceedings of the Sixth IEEE Real-Time Technology and Applications Symposium (RTAS '00), Washington - Brussels - Tokyo, pp. 124–133. IEEE, Los Alamitos (2000)
8. Hu, Y., Panda, B.: Identification of malicious transactions in database systems. In: IDEAS, pp. 329–335. IEEE Computer Society Press, Los Alamitos (2003)
9. Hu, Y., Panda, B.: A data mining approach for database intrusion detection. In: Haddad, H., Omicini, A., Wainwright, R.L., Liebrock, L.M. (eds.) SAC, pp. 711–716. ACM Press, New York (2004)
10. Mattsson, U.T.: A real-time intrusion prevention system for commercial enterprise databases. In: Ascenso, J., Belo, C., Vasiu, L., Saramago, M., Coelhas, H. (eds.) ICETE, pp. 275–280. INSTICC Press (2004)
11. Valeur, F., Mutz, D., Vigna, G.: A learning-based approach to the detection of SQL attacks. In: Julisch, K., Krügel, C. (eds.) DIMVA 2005. LNCS, vol. 3548, pp. 123–140. Springer, Heidelberg (2005)
12. Bertino, E., Kamra, A., Terzi, E., Vakali, A.: Intrusion detection in RBAC-administered databases. In: ACSAC, pp. 170–182. IEEE Computer Society Press, Los Alamitos (2005)
13. Spalka, A., Lehnhardt, J.: A comprehensive approach to anomaly detection in relational databases. In: Jajodia, S., Wijesekera, D. (eds.) Data and Applications Security XIX. LNCS, vol. 3654, pp. 207–221. Springer, Heidelberg (2005)
14. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic databases. In: VLDB, pp. 143–154. Morgan Kaufmann, San Francisco (2002)
15. Ryutov, T., Neuman, B.C., Kim, D., Zhou, L.: Integrated access control and intrusion detection for web servers. In: 23th International Conference on Distributed Computing Systems (23th ICDCS'2003). Providence, RI, pages 394-. IEEE Computer Society Press, Los Alamitos (2003)
16. Spitzner, L.: Honeypots: Catching the insider threat. In: ACSAC, pp. 170–181. IEEE Computer Society Press, Los Alamitos (2003)
17. Magklaras, G., Furnell, S.: Insider threat prediction tool: Evaluating the probability of IT misuse. Computers & Security 21(1), 62–73 (2002)
18. Heady, R., Luger, G., Maccabe, A., Servilla, M.: The architecture of a network level intrusion detection system. Technical report, University of New Mexico, Department of Computer Science, August (1990)
19. Ajith, S.P.: Intrusion detection systems using decision trees and support vector machines, URL: `citeseer.ist.psu.edu/741190.html`
20. Welz, M.G., Hutchison, A.: Interfacing trusted applications with intrusion detection systems. In: RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, pp. 37–53. Springer, London (2001)

21. Almgren, M., Lindqvist, U.: Application-integrated data collection for security monitoring. In: RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection, pp. 22–36. Springer, London, UK (2001)
22. Zamboni, D.: Data collection mechanisms for intrusion detection systems. Technical report(05 March, 2000)
23. Nyanchama, M., Osborn, S.: The role graph model and conflict of interest. ACM Transactions on Information and System Security 2(1), 3–33 (1999)
24. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 20(2), 38–47 (1996)
25. Modrakovic, M.: Reading and storing data directly from Oracle SGA using ProC*/C code (2004), URL: http://www.petefinnigan.com/Storing_Data_Directly_From_Oracle_SGA.pdf
26. Mattsson, U.: A practical implementation of a real-time intrusion prevention system for commercial enterprise databases. In: WSEAS, Copacabana, Rio de Janeiro, Brazil (2004)

Common Secure Index for Conjunctive Keyword-Based Retrieval over Encrypted Data

Peishun Wang, Huaxiong Wang, and Josef Pieprzyk

Department of Computing, Macquarie University
Sydney, NSW 2109, Australia
{pwang,hwang,josef}@ics.mq.edu.au

Abstract. We consider the following problem: users in a dynamic group store their encrypted documents on an untrusted server, and wish to retrieve documents containing some keywords without any loss of data confidentiality. In this paper, we investigate common secure indices which can make multi-users in a dynamic group to obtain securely the encrypted documents shared among the group members without re-encrypting them. We give a formal definition of common secure index for conjunctive keyword-based retrieval over encrypted data (CSI-CKR), define the security requirement for CSI-CKR, and construct a CSI-CKR based on dynamic accumulators, Paillier's cryptosystem and blind signatures. The security of proposed scheme is proved under strong RSA and co-DDH assumptions.

Keywords: Common secure index, search on encrypted data, Paillier's cryptosystem.

1 Introduction

Storages and backups of clients' data on remote untrusted servers are increasingly outsourced to data warehouses, and privacy invasion incidents about clients' sensitive data, such as information leakage caused by server administrator, has been increased more and more. The typical way to preserve clients' privacy is to store data in the form of ciphertext. It is important to build secure indices which enable a legitimate user to search encrypted data without decrypting them and revealing any information about the data to any other. In the literature, there are a number of research works on this topic [1, 5, 8, 9, 13], but the common setting of previous works is limited to a single-user. In realistic environments, however, secret data may be shared by a group, so group setting is more practical. Usually, the group may be dynamic, i.e., a person may join and leave the group, which makes the design more challenging. A naive solution will be to decrypt all data of the group, and re-encrypt them by a new group key. But this increases a computational overhead. Recently, Park *et al.* [11] proposed schemes for groups, which allow a user to obtain the encrypted documents in a secure way without re-encrypting them. In this paper, we study searchable common secure indices for groups and propose a new conjunctive keyword-based retrieval scheme for multi-users.

Related Work. Song, Wagner, and Perrig [13] first introduced the notion of searchable encryption. In their symmetric key setting, each word (or each pattern) of a document is encrypted separately and extra information is embedded into the ciphertext such that a search information (called trapdoor) can be provided to test whether a particular keyword is contained in a document. However, the scheme needs computation linear in the size of each document, fails to deal with compressed data, and is not secure against statistical analysis across encrypted data. These shortcomings were addressed by the subsequent work of Goh [8], who presented a scheme using Bloom filters to build a secure index for each file. Each keyword is processed using a pseudo-random function and then inserted into a Bloom filter. The trapdoor consists of an indicator of that which bits in the filter should be tested. Moreover, Goh formulated a security model for secure indices known as semantic security against adaptive chosen keyword attack (IND-CKA), which is the first formal notion of security defined for searchable encryption. But, Goh's scheme induces false positives inevitably and requires that all parties share all keys. Golle *et al.* [9] first constructed two schemes for conjunctive keyword search over encrypted data, which described three secure games that capture IND-CKA. Unfortunately, in their first scheme, the overhead transmission of the trapdoors is undesirable, and a large number (linear with respect to keywords) of pairing computations are requires to perform a search on every document; and their second construction is arguably unrealistic, particularly for a large collection of documents. Based on secret sharing and bilinear map for conjunctive keyword search, two more efficient schemes were proposed in [1], but the size of the trapdoors it generated is still linear in the number of documents being searched. Recently Boneh and Waters [5] developed a public-key encryption scheme for conjunctive keyword search from a generalization of anonymous identity based encryption. The scheme supports comparison queries (such as greater-than) and general subset queries. Nevertheless, the setting of all these schemes is limited to a single-user.

Based on Goh's scheme [8], Park *et al.* [11] proposed search schemes for groups, which can deal with membership changes without re-encrypting data. The main idea is to use one-way hash chain in reverse order to generate group session keys, encryption keys and index generation keys. In these schemes, all the group members use identical secret keys to make secure indices and trapdoors, and a set of new group keys must be generated for each session. Therefore, the schemes bring a risk to the key management; and the size of a query would become larger as the number of sessions increases. It should be noted that, if one of the leaving members reveals her group key to the server, the server can decrypt all the documents encrypted previously, because a user can know all of the previous group keys by hashing the current group key repeatedly. Additionally, these schemes are characterized by increased rate of false positives that are inherited from Bloom filters. To the best of our knowledge, it is the first work on searchable encryption in the multi-user setting. In this paper, we tackle all these concerns and develop a new searchable encryption for multi-users that we call Common

Secure Index for Conjunctive Keyword-based Retrieval over Encrypted Data (CSI-CKR).

Our Contributions. The contributions of this paper are four-fold: (1) We present a formal definition of CSI-CKR and describe the security requirement for CSI-CKR: Data Privacy against Server Administrators and Leaving Members and User Privacy against Insider and Group Manager, and the security models. (2) We construct a CSI-CKR from dynamic accumulators, Paillier's cryptosystem and blind signatures, and prove that the scheme satisfies the prescribed security requirement. (3) In our construction, authentication codes, common secure indices, trapdoors and encrypted data are all produced with public keys. This makes our scheme different from any previous searchable encryption in which the trapdoor is generated with secret keys. (4) Our work addresses all those shortcomings of Park *et al*'s schemes, and the proposed scheme achieves better performance.

Organization. Section 2 provides models and cryptographic preliminaries. In Section 3 we give an overview of RSA-based accumulators, Paillier's cryptosystem and blind signatures. In Section 4 we construct a CSI-CKR. Section 5 shows the correctness of the construction. In Section 6 the security of the proposed scheme is proved. In Section 7 we compare our scheme with other schemes. Finally Section 8 includes conclusions and future research.

2 Preliminaries

Notation. Through this paper, we use the following notation. Let $a \xleftarrow{R} A$ denote that an element a is chosen uniformly at random from the set A. $|G|$ stands for the cardinality of a group G. For any positive integer k, $[k]$ denotes the set of integers $\{1, \ldots, k\}$.

2.1 Model

We consider the group setting in which a collection of users stores their encrypted documents together with the document indices on an untrusted server. All the documents can be accessed by each user from the group. The group may be dynamic, i.e., a person may join and leave the group. For a leaving member, all documents stored on the server should be no longer accessible to the member including her own documents. A joining member not only can store her encrypted documents on the server that would be shared by other users, but also is able to retrieve all the previous and current documents that are owned by the group members.

There are three parts: a group manager (GM), users (members) in the group and a server. GM plays an important role as it manages the group members, their joining and leaving operations, group key generation and distribution, and all other operations related to maintenance of secret keys. First, GM setups the system and assigns an authentication code to each user. Then, every user encrypts her documents using the group encryption key, generates their corresponding common secure indices with the group public keys, which are the encryptions of principal keywords of the corresponding documents, and stores them on the server. When a user wishes to retrieve the documents containing some keywords, she uses the group public keys to make a trapdoor for the keywords and sends the trapdoor along with her authentication code to the server. Next, the server verifies the authentication code to confirm that the user is legitimate, and then, for the legitimate user, selects documents that contain the keywords by searching their common secure index with the trapdoor. After that, the server sends all matched data to the user. On receiving the data, the user encrypts them with her encryption key and sends the encrypted data to GM, who uses the group decryption key to decrypt the double-encrypted documents and return them to the user. Finally, the user gets the desired documents by decrypting her received data from GM with her decryption key.

Additionally, we assume that each document is associated with a list of keywords, and the number of keywords associated with a document (data) remains fixed. This constraint can be satisfied by simply adding null keywords to the list. As in [9], trapdoors specify which positions should be searched within an index.

Now we give a formal definition of Common Secure Index for Conjunctive Keyword-based Retrieval over Encrypted Data (CSI-CKR).

Definition 1. *A* **CSI-CKR** *consists of the following five components:*

SystemSetup *is to instantiate the scheme.*
It has one algorithm **SysSet**(k) *executed by GM, which takes as input a secure parameter s, and outputs a tuple of keys* (PK_g, PK_s, SK)*, where* PK_g *is published to the group,* PK_s *is sent to the server, and* SK *is kept secret by GM.*

AuthCodGen *is to generate group members' PIN numbers, their secure codes and a secure test code.*
It includes the following three algorithms:

GrpAut(G) *is executed by GM to make authentication codes for the original group. It takes as input all members* $M_1, \ldots, M_N$ *in the group* G*, outputs PIN numbers* d_i *and secure codes* c_i *for all members, and a secure test code STC for the server.*

MemJon$(\{M_{N+i}\}_{i=1,\ldots,r}, \{c_i\}_{i=1,\ldots,N})$ *is executed by GM interacting with old members to deal with the situation that some new users will join to the group. It takes as input all newly joining users* $\{M_{N+i}\}_{i=1,\ldots,r}$ *and all secure codes* $\{c_i\}_{i=1,\ldots,N}$ *of old members, outputs a PIN number* d_{N+i} *and a secure code* c_{N+i} *for each newly joining members, a new secure test code* STC' *for the server, and the updated secure code* c_i' *for every old member* M_i $(i = 1, \ldots, N)$*.*

MemLev$(\{d_{j_i}\}_{i=1,\ldots,r}, \{c_i\}_{i\in[N]\setminus\{j_1,\ldots,j_r\}})$ *is executed by GM interacting with current members (who are still in the group after some members leave) to deal with the situation that some members will leave the group.*

It takes as input all leaving members' PIN numbers $\{d_{j_i}\}_{i=1,\dots,r}$ and all current members' secure codes $\{c_i\}_{i\in[N]\setminus\{j_1,\dots,j_r\}}$, outputs a new secure test code STC' for the server, and the updated secure code c'_i for every current member M_i $(i = 1,\dots,N)$.

DataGen *is to build searchable encrypted data that are uploaded to the server. It includes the following two algorithms executed by group members:*

IndGen(R) *is to make a common secure index. It takes as input a data R, outputs its common secure index CSI_R.*

DatUpl(R, CSI_R) *is to upload the encrypted data with the common secure index to the server. It takes as input a data R and its common secure index CSI_R, and then uploads the encrypted data $S_g(R)$ with its CSI_R to the server.*

DataQuery *is to retrieve the encrypted data which contains specific keywords. It includes the following four algorithms:*

Trpdor(L', l) *is executed by a group member to make a trapdoor of a list of keywords the member wants to search. It takes as input a keyword list L' and the locations l of the keywords in the common secure index, outputs the trapdoor $T_{L'}$.*

MemChk(d_i, c_i, STC) *is executed by the server to check the membership. It takes as input the member's PIN number d_i and secure code c_i and the secure test code STC, outputs either* Yes *for access granted or* Access Denied *to terminate the scheme.*

SrhInd$(T_{L'}, CSI_R)$ *is executed by the server to scan all common secure indices against the trapdoor. If the output of* **MemChk**(d_i, c_i, STC) *is* Yes, *it takes as input a trapdoor $T_{L'}$ and a common secure index CSI_R, outputs an answer* Yes *or* No *to indicate that the data includes the searched keywords or not, respectively.*

DatDwn *is executed by the server to return a result to the member. Based on the output of* **SrhInd**$(T_{L'}, CSI_R)$, *it returns a collection of matched data or an answer* No Data Matched *to the member.*

DataDcrypt *is to obtain the data.*
It has one algorithm **DatDcp**$(S_g(R))$ *executed by a member interacting with GM, which takes as input an encrypted data $S_g(R)$, outputs the data R.*

2.2 Privacy

To a scheme of CSI-CKR, the attacker can be categorized into four classes:

Server Administrator (SA) - A person who has privileges to administer the server storing the encrypted data, but uses his administration rights in order to extract valuable information.

Leaving Member (LM) - A person who gained but now has lost access to the encrypted data, and tries to extract valuable information after his revocation.

Insider - A person who belongs to the group of trusted users, and tries to impersonate other identity to get information.

GM - A person who has all group secret keys and tries to find out about the information the document the user is interested in her query.

Depending on the types of attackers, the privacy requirements for CSI-CKR are as follows.

Data Privacy against SA - A SA can know nothing about keywords and document contents, i.e., encrypted documents must not provide any information about their contents, common secure indices leak nothing about their contents, and there must not exist information which is leaked from the query and the results generated in the search process.

Data Privacy against LM - A LM cannot search and retrieve any documents after she has left the group.

User Privacy against Insider - An insider cannot impersonate any other legitimate member in order to search and retrieve documents.

User Privacy against GM - A GM cannot know anything about the documents a user retrieves.

2.3 Security Models

We use two security models for our scheme. The first is the Semantic Security against Chosen Keyword Attacks (IND-CKA) introduced by Goh [8], which provides Data Privacy against SA. To simplify the proof, Golle *et al* [9] described three security games that capture IND-CKA and showed that they are asymptotically equivalent. We use one of these games, namely the Indistinguishability of Ciphertext from Random (ICR).

Because common secure indices and trapdoors concern the keyword list L only, and not the data R, for convenience, we use CSI_L instead of CSI_R. We assume that the number of keywords in a document is m.

Let Rand(L, I) is denoted a randomized keyword list formed from the keyword list $L = (w_1, \ldots, w_m)$ by replacing the keywords of L that are indexed by a subset $I \subset \{1, \ldots, m\}$ by random values. To describe conveniently, we use a notion of Distinguishing Trapdoor introduced in [9].

Definition 2. *A trapdoor T is distinguishable for keyword lists L_i and L_j if*

$$\mathbf{SrhInd}(T, CSI_{L_i}) \neq \mathbf{SrhInd}(T, CSI_{L_j}).$$

Given a set of indices $I \subset \{1, \ldots, m\}$, A trapdoor T distinguishes a keyword list L from Rand(L, I) if

$$\mathbf{SrhInd}(T, CSI_L) = Yes \text{ and } I \cap Loc(T) \neq \emptyset,$$

where $Loc(T)$ is the locations of the keywords of T in the keyword list.

Definition 3. ICR *is a game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:*

Setup *$\mathcal{A}$ adaptively selects a polynomial number of keyword lists, L, and requests the common secure indices, CSI_L, from $\mathcal{C}$.*

Queries *$\mathcal{A}$ may query $\mathcal{C}$ for the trapdoor $T_{L'}$ of a keyword list L'. With $T_{L'}$, $\mathcal{A}$ can invoke* $\mathbf{SrhInd}(T_{L'}, CSI_L)$ *on a common secure index CSI_L to determine if all keywords in the list L' are contained in L or not.*

Challenge *After making a polynomial number of queries, $\mathcal{A}$ decides on challenge by picking a keyword lists L_0 and a subset $I \subset \{1, \ldots, m\}$ such that $\mathcal{A}$ must not have asked for any trapdoor distinguishing L_0 from $L_1 = Rand(L_0, I)$, and sending them to $\mathcal{C}$. Then $\mathcal{C}$ chooses $b \xleftarrow{R} \{0, 1\}$, generates the common secure index CSI_{L_b}, and returns it to $\mathcal{A}$. After the challenge of determining b for $\mathcal{A}$ is issued, $\mathcal{A}$ is allowed again to query $\mathcal{C}$ with the restriction that $\mathcal{A}$ may not ask for the trapdoor that distinguishes L_0 from L_1.*

Response *Eventually $\mathcal{A}$ outputs a bit $b_\mathcal{A}$, and is successful if $b_\mathcal{A} = b$. The advantage of $\mathcal{A}$ in winning this game is defined as $Adv_\mathcal{A} = |Pr[b = b_\mathcal{A}] - 1/2|$, and the adversary is said to have an ϵ-advantage if $Adv_\mathcal{A} > \epsilon$.*

Now we introduce another notion of security that we call Unforgeability against Collaboration Member Attacks (UNF-CMA), which captures that the group membership is computationally unforgeable, and provides Data Privacy against LM and User Privacy against Insider.

Definition 4 (UNF-CMA). *A group membership authentication protocol is secure if it is hard for each collusion of up to t $(1 \leq t \leq |G|)$ members in the group G to forge a secure code of any other member (out of the collusion) in the group G or an outside user M' (out of the group) with a PIN number d' and a secure code c' which can prove that M' is a member of the group G.*

2.4 Complexity Assumptions

First let us state basic number-theoretic facts, and then make complexity assumptions that will be used in our work.

Fact: For a safe number $n = pq$, where $p = 2p' + 1, q = 2q' + 1$ and p, p', q, q' are primes, Euler's Totient function $\phi(n) = (p - 1)(q - 1)$, $\phi(n^2) = n\phi(n)$, and Carmichael's function $\lambda(n) = lcm(p - 1, q - 1)$.

Definition 5 (Strong RSA Assumption). *Given a safe number n and a random element $y \in Z_n^*$, it is infeasible to find any pair (x, e) $(e > 1)$ such that $y = x^e \bmod n$.*

We require a slightly stronger variant of Decision Diffie-Hellman (DDH) assumption – co-DDH, which originated in [4] and was recently used as a hardness assumption in [1].

Definition 6 (co-DDH Assumption). *Let $G_0 = \langle g \rangle$ and $G_1 = \langle P \rangle$ be two cyclic groups, and the DDH assumption holds within G_1. Given random $g^b \in G_0$ and $P, aP, bP, cP \in G_1$, a probabilistic polynomial time algorithm $\mathcal{A}$ is said to have an ϵ-advantage to break the co-DDH assumption if*

$$|Pr[\mathcal{A}(g^b, P, aP, bP, abP) = 1] - Pr[\mathcal{A}(g^b, P, aP, bP, cP) = 1]| > \epsilon.$$

3 Overview of Techniques

3.1 RSA-Based Accumulators

The basic RSA-based accumulator [3] is constructed as follows: for a set of elements $X = \{x_1, \ldots, x_m\}$, the accumulator function is

$$y_i = f(y_{i-1}, x_i),$$

where f is a one-way function: $f(u, x) = u^x \bmod n$ for suitably-chosen values of the seed u and RSA modulus n.

The accumulated value is

$$v = u^{x_1 \ldots x_m} \bmod n,$$

and the witness for the element x_i is

$$w_i = u^{x_1 \ldots x_{i-1} x_{i+1} \ldots x_m} \bmod n.$$

Baric and Pfitzmann [2] constrained the elements to be primes and showed that the accumulator is collision-resistant assuming factoring is difficult, that means it is hard to compute a witness for an element that is not accumulated.

Camenisch and Lysyanskaya [6] further developed dynamic accumulators, which feature that the accumulator can dynamically add or delete elements into/from the original set. In their setting, the domain of the elements consists of prime numbers in particular range, the seed u is from the group of quadratic residues.

3.2 Paillier's Cryptosystem

We now briefly review Paillier's Cryptosystem. For the detail, refer to [10]. Let n be a safe number. $\mathcal{B}_\alpha \subset Z^*_{n^2}$ denotes the set of elements of order $n\alpha$, and $\mathcal{B}$ denotes their disjoint union for $\alpha = 1, \ldots, \lambda$, where λ is adopted instead of $\lambda(n)$ for visual comfort. Randomly select a base g from $\mathcal{B}$, and define $L(x) = \frac{x-1}{n}$.

Paillier's Cryptosystem defines an integer-valued bijective function:

$$E_g \colon Z_n \times Z^*_n \rightarrow Z^*_{n^2},\ (x, r) \rightarrow g^x \cdot r^n \ \mathsf{mod}\ n^2,$$

where (n, g) are public parameters whilst the pair (p, q) (or equivalently λ) remains private. The encryption and decryption algorithms are as follows:

Encryption:
plaintext $x < n$, randomly select $r < n$, ciphertext $y = g^x \cdot r^n \ \mathsf{mod}\ n^2$.
Decryption:
ciphertext $y < n^2$, plaintext $x = D(y) = \frac{L(y^\lambda \ \mathsf{mod}\ n^2)}{L(g^\lambda \ \mathsf{mod}\ n^2)} \ \mathsf{mod}\ n$.

It has additive homomorphic properties:

$$\forall \sigma \in Z^+,\ \ D(y^\sigma \ \mathsf{mod}\ n^2) = \sigma x \ \mathsf{mod}\ n, \qquad \text{and}$$
$$\forall y_1, y_2 \in Z_{n^2},\ D(y_1 y_2 \ \mathsf{mod}\ n^2) = x_1 + x_2 \ \mathsf{mod}\ n$$

3.3 Blind Signatures

Blind signature schemes, first introduced by Chaum in [7], allow a user to get a signature without giving the signer any information about the actual message. The signer has a signing function S' and its inverse S, where S' is known only to the signer (secret key) and S is publicly known (public key), such that $S(S'(x)) = x$ for a message x. The user has a commutating function C_M and its inverse C'_M, where both are known only to the user, such that $C'_M(S'(C_M(x))) = S'(x)$, and $C_M(x)$ and $S'(x)$ give no clue about x. Briefly, the protocol is precessed as follows:

1. The user chooses x at random such that $H(x)$, where H is a public hash function, and supplies $C_M(x)$ to the signer.
2. The signer signs $C_M(x)$ by applying S' and returns the signed matter $S'(C_M(x))$ to the user;
3. The user strips signed matter by application of C'_M, yielding $C'_M(S'(C_M(x))) = S'(x)$;
4. Anyone can check that the stripped matter S' was formed by the signer, by applying the signer's public key S and checking that $H(S(S'(x)))$.

The signer knows nothing about the correspondence between the elements of the set of stripped signed matter $S'(x_i)$ and the element of the set of unstripped signed matter $S'(C_M(x_i))$.

4 Constructing CSI-CKR

4.1 SystemSetup – *System Instantiation Process*

SysSet(k):

1. Takes a security parameter k, chooses a random RSA modulus n of length k. Let $f : \{0,1\}^k \times \{0,1\}^l \rightarrow Z_n$ be a pseudorandom function, where l is the length of the longest word. We denote $f(\cdot,\cdot)$ by $f(\cdot)$ for convenience.
2. Chooses a random quadratic residue u modulo n $(u \neq 1)$, and defines $\mathcal{M} = \{e \in primes : e \neq p', q' \wedge A \leq e \leq B\}$, where $2 < A$ and $B < A^2$ and A, B are dependent on k and the upper bound of $|G|$.
3. Chooses $\sigma \overset{R}{\longleftarrow} Z^+$, and computes $\beta = \sigma\lambda \bmod \phi(n^2)$ and $\gamma = \sigma L(g^\lambda \bmod n^2) \bmod n$, where λ, g, and $L(\cdot)$ are parameters in Paillier's cryptosystem.
4. Chooses a cyclic group $G_1 = \langle P \rangle$, in which the computation is based on the modulus n and the DDH assumption holds.
5. Chooses a pair of Blind Signature keys for the group: signing function S_g and its inverse S'_g, where S_g is for group members to encrypt their data.
6. Publishes $PK_g = (g, \gamma, f(\cdot), P, n, S_g)$ to the group (group public key), sends $PK_s = (\mathcal{M}, \beta, L(\cdot), n)$ to the server, and keeps $SK = (\sigma, \lambda, S'_g)$ privately (group secret key).

4.2 AuthCodGen – *Group Authentication Process*

GrpAut(G): The group G has N members $\{M_1, \ldots, M_N\}$

1. For every member M_i $(1 \leq i \leq N)$, GM selects $d_i \overset{R}{\longleftarrow} \mathcal{M}$ as M_i's PIN number.

2. GM picks up $d^* \overset{R}{\longleftarrow} \mathcal{M} \setminus D$, where $D = \{d_1, \ldots, d_N\}$, and computes M_i's secure code
$$c_i = u^{d_1 \cdots d_{i-1} d_{i+1} \cdots d_N d^*} \bmod n.$$
3. GM sends the server the secure test code
$$STC = u^{d_1 \cdots d_{N+1} d^*} \bmod n,$$
and puts d^* in D.

MemJon($\{M_{N+i}\}_{i=1,\ldots,r}, \{c_i\}_{i=1,\ldots,N}$): New users $M_{N+1}, \ldots, M_{N+r}$ $(0 < r \leq |\mathcal{M}| - |D|)$ would join to the group G.
1. GM selects $r+1$ random numbers $\{d_{N+i}\}_{i=1,\ldots,r}$ and d^*_+ from $\mathcal{M} \setminus D$, and computes
$$v = d_{N+1} \cdots d_{N+r} d^*_+ \bmod \phi(n),$$
then sends v to every old member.
2. GM assigns $\{d_{N+i}\}_{r=1,\ldots,r}$ to $\{M_{N+i}\}_{i=1,\ldots,r}$ respectively as their PIN numbers, and computes their secure codes
$$c_i = STC^{v d_{N+i}^{-1}} \bmod n \ (i = 1, \ldots, r)$$
and sends the server the new secure test code
$$STC' = STC^v \bmod n,$$
then puts $\{d_{N+i}\}_{r=1,\ldots,r}$ and d^*_+ in D.
3. When old members $\{M_1, \ldots, M_N\}$ receive v, they update their secure codes as follows:
$$c'_i = (c_i)^v \bmod n, \ i = 1, \ldots, N.$$
4. On receiving the new secure test code STC', the server replaces STC with STC'.

MemLev($\{d_{j_i}\}_{i=1,\ldots,r}, \{c_i\}_{i \in [N] \setminus \{j_1, \ldots, j_r\}}$): Old members$M_{j_1}, \ldots, M_{j_r}$ $(0 < r < |G|)$ would leave from the group G.
1. GM computes
$$v = d_{j_1} \cdots d_{j_r} \bmod \phi(n)$$
and the new secure test code
$$STC' = STC^{v^{-1}} \bmod n,$$
and then sends STC' to the server and v and STC' to all members who are still in G after $\{M_{j_i}\}_{i=1,\ldots,r}$ leave. Finally GM deletes $\{d_{j_i}\}_{i=1,\ldots,r}$ from D.
2. When a current member M_i receives v and STC', she computes the integers a, b such that $ad_i + bv = 1$, and then updates her secure codes as follows:
$$c'_i = (c_i)^b (STC')^a \bmod n.$$
3. On receiving the new secure test code STC', the server replaces STC with STC'.

4.3 DataGen – *Data Build Process*

IndGen(R): A member of G chooses $\rho \overset{R}{\longleftarrow} Z^+$. Lets $s_0 = \rho P$, and for each word w_j in her data R $(j = 1, \ldots, m)$, computes $s_j = \rho\gamma f(w_j) P$. Then outputs $CSI_R = (s_0, s_1, \ldots, s_m)$ as the common secure index of R.

DatUpl(R, CSI_R): The member uses group's public key S_g to encrypted her Data $S_g(R)$, and uploads the encrypted data $S_g(R)$ with its CSI_R to the server.

4.4 DataQuery – *Data Search and Download Process*

Trpdor(L', l): A member chooses $r_i \xleftarrow{R} Z_n$. For each word $w'_j \in L'_i$, let $c_j = g^{f(w_j)} \cdot {r_i}^n \bmod n^2$. The member computes $C = \prod_{j=1}^{k} c_j \bmod n^2$. Then the member takes $T_{L'} = (C, l)$ as the trapdoor of the keyword list L', where $l = (l_1, \ldots, l_k)$ is the set of the locations of keywords $(w'_1, \ldots, w'_k)$ in the index, and sends the server the trapdoor $T_{L'}$ along with her PIN number d_i and secure code c_i.

MemChk(d_i, c_i, STC): The server verifies if $d_i \in \mathcal{M}$ and $(c_i)^{d_i} = STC \bmod n$. If so, outputs Yes; otherwise, returns the member Access Denied and terminates the scheme.

SrhInd($T_{L'}, CSI_R$): If the output of **MemChk(d_i, c_i, STC)** is Yes, for a CSI_R, the server tests $L(C^{\beta s_0} \bmod n^2) \equiv \sum_{j \in l} s_j \pmod{n}$. If so, outputs Yes; otherwise, outputs No.

DatDwn: If the output of **SrhInd($T_{L'}, CSI_R$)** is Yes, puts the data R in a collection, then check the next common secure index; otherwise, checks the next common secure index. Finally, if the collection is not empty, the server returns the collection to the member; otherwise, returns No Data Matched to the member.

4.5 DataDcrypt – *Data Decryption Process*

DatDcp($S_g(R)$): When the member receives a encrypted data $S_g(R)$, she uses her Blind Signature commutating function C_M and its inverse C'_M to interact with GM as follows:

1. The member uses C_M to encrypt $S_g(R)$ and sends $C_M(S_g(R))$ to GM.
2. GM uses the secret key S'_g to decrypt

$$S'_g(C_M(S_g(R))) = C_M(S'_g(S_g(R))) = C_M(R)$$

   and returns it to the member.
3. The member uses C'_M to decrypt $C_M(R)$ to get the data R.

5 Correctness

5.1 Correctness of Authentication

For the outputs of algorithms **GrpAut(G)** and **MemJon($\{M_{N+i}\}_{i=1,\ldots,r}, \{c_i\}_{i=1,\ldots,N}$)**, it is very simple to prove that the output of algorithm **MemChk(d_i, c_i, STC)** is correct.

Considering the outputs of algorithm **MemLev($\{d_{j_i}\}_{i=1,\ldots,r}, \{c_i\}_{i \in [N] \setminus \{j_1,\ldots,j_r\}}$)**, let's verify the correctness of the result of algorithm **MemChk(d_i, c_i, STC)**.

$$
\begin{aligned}
(c_i)^{d_i} \bmod n &= ((c_i)^b (STC')^a)^{d_i} \bmod n \\
&= (((c_i)^b (STC')^a)^{d_i v})^{v^{-1}} \bmod n \\
&= (((c_i)^{d_i})^{bv} ((STC')^v)^{a d_i})^{v^{-1}} \bmod n \\
&= (STC^{bv} STC^{a d_i})^{v^{-1}} \bmod n \\
&= (STC^{a d_i + bv})^{v^{-1}} \bmod n \\
&= STC' \bmod n
\end{aligned}
$$

So, the result of authentication is correct.

5.2 Correctness of Search

From the properties of Paillier's Cryptosystem, we have

$$t(m_1 + \ldots + m_k) \equiv \frac{L((c_1 \ldots c_k)^{t\lambda} \bmod n^2)}{L(g^\lambda \bmod n^2)} \pmod n.$$

Thus

$$t(m_1 + \ldots + m_k) L(g^\lambda \bmod n^2) \equiv L((c_1 \ldots c_k)^{t\lambda} \bmod n^2) \pmod n.$$

Let $t = \rho\sigma P$ and $m_j = f(w_j)$, so we have:

$$\textstyle\sum_{j \in l} s_j \equiv L(C^{\beta s_0} \bmod n^2) \pmod n.$$

Therefore, the result of search is correct.

6 Security

Theorem 1. *The proposed CSI-CKR scheme is semantically secure under the security game ICR if co-DDH is intractable.*

Proof. Suppose that the scheme is not semantically secure under the security game ICR. Then there exists an adversary $\mathcal{A}$ that wins the ICR game with an ϵ-advantage. We build an adversary $\mathcal{A}'$ that uses $\mathcal{A}$ as a subroutine and breaks the co-DDH assumption with the same ϵ-advantage. The running time of $\mathcal{A}'$ is approximately the same as $\mathcal{A}$'s.

Let (g^b, P, aP, bP, cP) be $\mathcal{A}'$'s co-DDH challenge. $\mathcal{A}'$'s goal is to break the co-DDH assumption, or in other words to decide whether $c = ab$. $\mathcal{A}'$ works by interacting with $\mathcal{A}$ in the ICR game as follows:

Setup: $\mathcal{A}$ makes a polynomial number of requests for common secure indices, which $\mathcal{A}'$ answers as follows. Let one of $\mathcal{A}$'s keyword lists be $L_i = (w_{i,1}, \ldots, w_{i,m})$. $\mathcal{A}'$ chooses a value $\rho \xleftarrow{R} Z^+$, compute $s_0 = \rho P$. For each word $w_{i,j} \in L_i$ $(1 \le j \le m)$, it picks a value $x_{i,j} \xleftarrow{R} Z_n$, and compute $s_j = \rho\gamma x_{i,j} bP$. Note that $\mathcal{A}'$ is given P and bP as part of co-DDH challenge, so it can compute s_0 and s_j. To be consistent across different queries, $\mathcal{A}'$ keeps track of the corresponding pair $(w_{i,j}, x_{i,j})$. Finally, it returns the common secure index $CSI_{L_i} = (s_0, s_1, \ldots, s_m)$ to $\mathcal{A}$.

Queries: When $\mathcal{A}$ makes a trapdoor query on keyword list $L' = (w'_1, \ldots, w'_k)$ with keywords' locations $l = (l_1, \ldots, l_k)$ in common secure indices, $\mathcal{A}'$ chooses a value $r \xleftarrow{R} Z_n$, and computes $c_j = (g^b)^{y_j} \cdot r^n \bmod n^2$ for each word $w'_j \in L'$, where $y_j = x_{i,j}$ if w'_j previously appeared in any one of $\mathcal{A}$'s queries to either its **IndGen**(R) or **Trpdor**(L', l) oracles or $y_j \xleftarrow{R} Z_n$ otherwise (also the corresponding pair (w'_j, y_j) has to be kept in memory for future use). Observe that, although $\mathcal{A}'$ does not know b, it can compute c_j, because g^b is given to it as part of its co-DDH challenge. Then $\mathcal{A}'$ computes $C = \prod_{j=1}^{k} c_j \bmod n^2$. Finally, $\mathcal{A}'$ returns the trapdoor $T_{L'} = (C, l)$ to $\mathcal{A}$. Because $\mathcal{A}'$ consistently uses the same value $x_{i,j}$ for word w_j, $T_{L'}$ is a valid trapdoor for $L' = (w'_1, \ldots, w'_k)$, and **SrhInd**$(T_{L'}, CSI_{L_i})$ returns Yes if and only if L_i contains all the keywords in L' at the locations specified by $l = (l_1, \ldots, l_k)$.

Challenge: After making polynomially many index and trapdoor queries, $\mathcal{A}$ decides on a challenge by submitting the challenge keyword list $L_\delta = (w_{\delta,1}, \ldots, w_{\delta,m})$. Then $\mathcal{A}'$ computes $s_0 = \rho a P$ and $s_j = \rho\gamma y_{\delta,j} c P$, where $y_{\delta,j} = x_{i,j}$ if $w_{\delta,j}$ previously appeared in one of $\mathcal{A}$'s queries (including index queries and trapdoor ones) or $y_{\delta,j} \xleftarrow{R} Z_n$ otherwise. Notice that $\mathcal{A}'$ can compute s_0 and s_j by using given aP and cP. Finally $\mathcal{A}'$ returns to $\mathcal{A}$ the challenge common secure index $CSI_{L_\delta} = (s_0, s_1, \ldots, s_m)$. Note that, if $c = ab$, CSI_{L_δ} is a correct common secure index for L_δ; if $c \neq ab$, CSI_{L_δ} is a correct common secure index for some other arbitrary keyword list. After the challenge, $\mathcal{A}$ is allowed to make more trapdoor and index queries with the requested restriction, which $\mathcal{A}'$ answers as it did in the stage of **Queries**.

Response: $\mathcal{A}$ outputs a guess that represents its decision as to whether CSI_{L_δ} is a correct common secure index for L_δ or not. Then $\mathcal{A}'$ returns the same guess as its own answer to its co-DDH challenge.

Since CSI_{L_δ} is a correct common secure index for L_δ if and only if $c = ab$, it follows that the advantage of $\mathcal{A}'$ in breaking co-DDH assumption is ϵ if $\mathcal{A}$ has an ϵ-advantage in breaking the ICR game.

Theorem 2. *The proposed CSI-CKR scheme is secure under UNF-CMA if Strong RSA assumption holds.*

Proof. In our construction, all LMs and Insiders (members) only use group public keys to build common secure indices and trapdoors, and Insiders only use group public keys to update their secure codes, they never know the secret keys.

First we prove that it is hard for each collusion of up to t $(1 \leq t \leq |G|)$ members in the group G to forge an outside user M' (out of the group) with a PIN number $d' \in \mathcal{M} \setminus ID_G$ and a secure code c' which can prove that M' is a member of the group G, where ID_G is the collection of PIN numbers of all members in G.

Without loss of generality, assume $t = |G|$. According to Camenisch *et al*'s theorem (Theorem 2 in their paper [6]), it is hard for $|G|$ members to collaborate to forge a legitimate member M' with a PIN number $d' \in \mathcal{M} \setminus D$ and a secure code c' which can make the output of algorithm **MemChk**(d_i, c_i, STC) to be Yes.

All the numbers in $D \setminus ID_G$ are chosen uniformly at random by GM, and there exists no member in G who knows these numbers. So let

$$x = u^{(\prod_{d^* \in D \setminus ID_G} d^*)} \bmod n,$$

all the secure codes of the members seem to be computed on x. According to Baric *et al*'s theorem (Theorem 5 in their paper [2]), it is hard for $|G|$ members to collaborate to forge a legitimate member M' with a PIN number $d' \in D \setminus ID_G$ and a secure code c' which can make the output of algorithm **MemChk(**d_i, c_i, STC**)** to be Yes.

Next we show that it is computationally infeasible for each collusion of up to t ($1 \leq t < |G|$) members in the group G to forge a secure code of any other member (out of the collusion) in the group.

We suppose that a collusion of t ($1 \leq t < |G|$) members $\{M_{j_1}, \ldots, M_{j_t}\}$ in the group G can forge a secure code c' of a member $M' \in G \setminus \{M_{j_1}, \ldots, M_{j_t}\}$ whose PIN number is d'. This means,

$$(c')^{d'} \equiv STC \equiv y^r \pmod{n},$$

where $y = u^{(\prod_{d \in D \setminus \{d_{j_1} \cdots d_{j_t}\}} d)} \bmod n$ and $r = d_{j_1} \cdots d_{j_t}$.

Now let's construct the d'-th root x of y as in [12]:

Since d' is prime, with the extended Euclidean algorithm, we can compute $a, b \in Z$ such that $ar + bd' = 1$. Let $x = (c')^a y^b$, we have

$$\begin{aligned} x^{d'} \bmod n &= ((c')^a y^b)^{d'} \bmod n \\ &= ((c')^{d'})^a y^{bd'} \bmod n \\ &= STC^a y^{bd'} \bmod n \\ &= y^{ar} y^{bd'} \bmod n \\ &= y \bmod n. \end{aligned}$$

Thus, we break the Strong RSA assumption.

Because the algorithm **DatDcp(**$S_g(R)$**)** follows from Blind Signature schemes in a straightforward way, we have the following theorem immediately.

Theorem 3. *The proposed CSI-CKR scheme is secure for User Privacy against GM.*

7 Comparison with Park *et al*'s Schemes

In this section, we compare our construction with Park *et al*'s schemes. As we mentioned above, their schemes use an identical group session key as their authentication codes for all group members, so it cannot provide User Privacy against Insider. Their schemes use Goh's single-user scheme [8] to build common secure indices and trapdoors, and use the group encryption and decryption keys to process the data, that means, every member knows the secret keys, hence, their schemes bring a big risk to the key management; and our scheme is based on a new idea different from any previous single-user schemes, and only uses public keys to generate authentication codes, build common secure indices and trapdoors, and encrypt the data. Therefore, our scheme is better to key management. After the q-th session in their schemes, a user must make q trapdoors for a list of keywords, thus, when the q is big enough, their schemes become much inefficient. On the other hand, the size of trapdoor in our scheme is fixed to $2n + \log m$. So our scheme is more practical than theirs. In their schemes, if a LM reveals the group decryption key to a SA, the SA can decrypt all the documents encrypted previously, as a user can know all of the previous group encryption keys by hashing the current group encryption key repeatedly. This breaks their schemes down completely. However, in our scheme, the members do not have the decryption key, so our scheme avoids such attacks. In addition, our scheme does not induce any false positives that their schemes bring inevitably.

8 Conclusion and Open Problem

Our proposed scheme, CSI-CKR, is the searchable protocol on multi-users shared encrypted data. We present a formal definition of CSI-CKR, and defined the security requirement for the dynamic group retrieval system on encrypted data. Our scheme can provide Data Privacy against SA and LM and User Privacy against Insider and GM. Our scheme introduced GM to play a trusted third part, so designing the scheme for group retrieval without GM is still a challenging problem.

References

[1] Ballard, L., Kamara, S., Monrose, F.: Achieving Efficient Conjunctive Keyword Searches over Encrypted Data. In: Qing, S., Mao, W., Lopez, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005)

[2] Baric, N., Pfitzmann, B.: Collision-free accumulators and fail-stop signature schemes without trees. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 480–494. Springer, Heidelberg (1997)

[3] Benaloh, J., de Mare, M.: One-way accumulators: a decentralized alternative to digital signatures. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 274–285. Springer, Heidelberg (1994)

[4] Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. SIAM Journal of Computing 32(3), 586–615 (2003)

[5] Boneh, D., Waters, B.: Conjunctive, Subset, and Range Queries on Encrypted Data. Cryptology ePrint Archive, Report (2006)/287

[6] Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)

[7] Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology: CRYPTO 1982, pp. 199–203. Plenum Publishing, New York (1982)

[8] Goh, E.-J.: Secure indexes. In: Cryptology ePrint Archive, Report, /216, (February 25, 2004) See http://eprint.iacr.org/2003/216/forthelatestversion

[9] Golle, P., Staddon, J., Waters, B.: Secure Conjunctive Search over Encrypted Data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
[10] Paillier, P.: Public-Key Cryptosystems based on Composite Degree Residue Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
[11] Park, H.A., Byun, J.W., Lee, D.H.: Secure Index Search for Groups. In: Katsikas, S.K., Lopez, J., Pernul, G. (eds.) TrustBus 2005. LNCS, vol. 3592, pp. 128–140. Springer, Heidelberg (2005)
[12] Shamir, A.: On the Generation of Cryptographically Strong Pseudorandom Sequences. ACM Transaction on Computer Systems 1(1), 38–44 (1983)
[13] Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 44–55 (May 2000)

Generating Microdata with *P*-Sensitive *K*-Anonymity Property

Traian Marius Truta[1], Alina Campan[2], and Paul Meyer[1]

[1] Department of Computer Science, Northern Kentucky University,
Highland Heights, KY 41099, USA
{trutat1,meyerp1}@nku.edu
[2] Department of Computer Science, Babes-Bolyai University,
Cluj-Napoca, RO-400084, Romania
alina@cs.ubbcluj.ro

Abstract. Existing privacy regulations together with large amounts of available data have created a huge interest in data privacy research. A main research direction is built around the *k*-anonymity property. Several shortcomings of the *k*-anonymity model have been fixed by new privacy models such as *p*-sensitive *k*-anonymity, *l*-diversity, (α, *k*)-anonymity, and *t*-closeness. In this paper we introduce the *EnhancedPKClustering* algorithm for generating *p*-sensitive *k*-anonymous microdata based on frequency distribution of sensitive attribute values. The *p*-sensitive *k*-anonymity model and its enhancement, extended *p*-sensitive *k*-anonymity, are described, their properties are presented, and two diversity measures are introduced. Our experiments have shown that the proposed algorithm improves several cost measures over existing algorithms.

Keywords: Privacy, *k*-anonymity, *p*-sensitive *k*-anonymity, attribute disclosure.

1 Introduction

The increased availability of individual data has nowadays created a major privacy concern. Legislators from many countries have tried to regulate the use and disclosure of confidential information (or data) [2]. New privacy regulations, such as the *Health Insurance Portability and Accountability Act* (*HIPAA*) [7], along with the necessity of collecting personal information have generated a growing interest in privacy research. Several techniques that aim to avoid the disclosure of confidential information by processing sensitive data before public release have been presented in the literature. Among them, the *k*-anonymity model was recently introduced [16, 17]. This property requires that in the *released* (a.k.a. *masked*) *microdata* (datasets where each tuple belongs to an individual entity, e.g. a person, a company) every tuple will be indistinguishable from at least (*k*-1) other tuples with respect to a subset of attributes called *key* or *quasi-identifier* attributes.

Although the model's properties, and the techniques used to enforce it on data, have been extensively studied [1, 4, 11, 16, 18, 20, etc.], recent results have shown that *k*-anonymity fails to protect the privacy of individuals in all situations [14, 19, 23, etc.]. New enhanced privacy models have been proposed in the literature to deal with *k*-anonymity's limitations with respect to *sensitive attributes disclosure* (this term will be explained in the next section). These models follow one of the following two approaches: the *universal* approach uses the same privacy constraints for all individual entities, while the *personalized* approach allows users or data owners to customize the amount of privacy they need. The first category of privacy protection models, based on the universal approach, includes: *p*-sensitive *k*-anonymity [19] with its extension called extended *p*-sensitive *k*-anonymity [5], *l*-diversity [14], (α, *k*)-anonymity [22], and *t*-closeness [13]. The only personalized privacy protection model we are aware of is personalized anonymity [23].

In this paper we introduce an efficient algorithm for anonymizing a microdata set such that its released version will satisfy *p*-sensitive *k*-anonymity. Our main interest in developing a new anonymization algorithm was to obtain better *p*-sensitive *k*-anonymous solutions w.r.t. various cost measures than the existing algorithms by taking advantage of the known properties of the *p*-sensitive *k*-anonymity model.

In order to describe the algorithm, the *p*-sensitive *k*-anonymity model, extended *p*-sensitive *k*-anonymity model, and their properties are presented. Along with existing cost measures such as *discernability measure (DM)* [3] and *normalized average cluster size metric* (*AVG*) [12], two diversity measures are introduced. The proposed algorithm is based on initial microdata frequency distribution of sensitive attribute values. It partitions an initial microdata set into clusters using the properties of the *p*-sensitive *k*-anonymity model. The released microdata set is formed by generalizing the quasi-identifier attributes of all tuples inside each cluster to the same values. We compare the results obtained by our algorithm with the results of those from both the *Incognito* algorithm [11], which was adapted to generate *p*-sensitive *k*-anonymous microdata, and the *GreedyPKClustering* algorithm [6].

The paper is structured as follows. Section 2 presents the *p*-sensitive *k*-anonymity model along with its extension. Section 3 introduces the *EnhancedPKClustering* algorithm. Experimental results and conclusions are presented in Sections 4 and 5.

2 Privacy Models

2.1 *p*-Sensitive *k*-Anonymity Model

The *p*-sensitive *k*-anonymity model is a natural extension of *k*-anonymity that avoids several shortcomings of this model [19]. Next, we present these two models.

A microdata is a set of tuples in the relational sense. The initial dataset (called initial microdata and labeled $\mathcal{IM}$) is described by a set of attributes that are classified into the following three categories:

- I_1, I_2,..., I_m are *identifier* attributes such as *Name* and *SSN* that can be used to identify a record.
- K_1, K_2,…, K_q are *key* or *quasi-identifier* attributes such as *ZipCode* and *Sex* that may be known by an intruder.
- S_1, S_2,…, S_r are *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 [21]. To avoid this possibility of disclosure, one frequently used solution is to further process (modify) the initial microdata through generalization and suppression 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.

There is no consensus in the literature over the term used to denote a *QI*-cluster. This term was not defined when *k*-anonymity was introduced [16, 17]. More recent papers use different terminologies such as *equivalence class* [22] and *QI-group* [23].

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 does not provide the amount of confidentiality required for every individual [14, 19, 22]. To briefly justify this affirmation, we distinguish between two possible types of disclosure; namely, *identity disclosure* and *attribute disclosure*. *Identity disclosure* refers to re-identification of an entity (person, institution) and *attribute disclosure* occurs when the intruder finds out something new about the target entity [10]. *K*-anonymity protects against identity disclosure but fails to protect against attribute disclosure when all tuples of a *QI*-cluster share the same value for one sensitive attribute [19]. This attack is called *homogeneity attack* [14] and can be avoided by enforcing a more powerful anonymity model than *k*-anonymity, for example *p*-sensitive *k*-anonymity. A different type of attack, called *background attack*, is presented in [14]. In this attack, the intruder uses background information that allows him / her to rule out some possible values of the sensitive attributes for specific individuals. Protection against background attacks is more difficult since the data owner is unaware of the type of background knowledge an intruder may posses. To solve this problem particular assumptions should be made, and anonymization techniques by themselves will not fully eliminate the risk of the background attack [22]. Still, enhanced anonymization techniques try to perform as well as possible in case of background attacks.

The *p*-sensitive *k*-anonymity model considers several sensitive attributes that must be protected against attribute disclosure. Although initially designed to protect against homogeneity attacks, it also performs well against different types of background attacks. It has the advantage of simplicity and allows the data owner to customize the desired protection level by setting various values for *p* and *k*. Intuitively, the larger the parameter *p*, the better is the protection against both types of attacks.

Definition 3 (***p-sensitive k-anonymity property***)**:** A $\mathcal{MM}$ satisfies ***p-sensitive k-anonymity property*** if it satisfies *k*-anonymity and the number of distinct attributes for each confidential attribute is at least *p* within the same *QI*-cluster from the $\mathcal{MM}$.

To illustrate this property, we consider the masked microdata from Table 1 where *Age* and *ZipCode* are quasi-identifier attributes, and *Diagnosis* and *Income* are confidential attributes:

Table 1. Masked microdata example for *p*-sensitive *k*-anonymity property

Age	**ZipCode**	**Diagnosis**	**Income**
20	41099	AIDS	60,000
20	41099	AIDS	60,000
20	41099	AIDS	40,000
30	41099	Diabetes	50,000
30	41099	Diabetes	40,000
30	41099	Tuberculosis	50,000
30	41099	Tuberculosis	40,000

The above masked microdata satisfies 3-anonymity property with respect to *Age* and *ZipCode*. To determine the value of *p*, we analyze each *QI*-cluster with respect to their confidential attribute values. The first *QI*-cluster (the first three tuples in Table 1) has two different incomes (*60,000* and *40,000*), and only one diagnosis (*AIDS*), therefore the highest value of *p* for which *p*-sensitive 3-anonymity holds is 1. As a result, a presumptive intruder who searches information about a young person in his twenties that lives in zip code area 41099 will discover that the target entity suffers from *AIDS*, even if he doesn't know which tuple in the first *QI*-cluster corresponds to that person. This attribute disclosure problem can be avoided if one of the tuples from the first *QI*-cluster would have a value other than *AIDS* for *Diagnosis* attribute. In this case, both *QI*-clusters would have two different illnesses and two different incomes, and, as a result, the highest value of *p* would be 2.

From the definitions of *k*-anonymity and *p*-sensitive *k*-anonymity models we easily infer that 2-sensitivity 2-anonymity is a necessary condition to protect any masked microdata against any type of disclosure, identity or attribute disclosure. Unfortunately, the danger of disclosure is not completely eliminated since an intruder may "guess" the identity or attribute value of some individuals with a probability of ½. For many masked microdata such a high probability is unacceptable, and the values of *k* and/or *p* must be increased.

2.2 *p*-Sensitive *k*-Anonymity Model Properties

We introduce the following notations, which will be used for expressing several properties of *p*-sensitive *k*-anonymity and for presenting our anonymization algorithm. For any given microdata set $\mathcal{M}$, we denote by:

- n – the *number* of tuples in $\mathcal{M}$.
- r – the number of confidential attributes in $\mathcal{M}$.
- s_j – the number of distinct values for the confidential attribute S_j $(1 \leq j \leq r)$.

- v_i^j – the distinct values for the confidential attribute S_j in descending order of their occurrences ($1 \leq j \leq r$ and $1 \leq i \leq s_j$).
- f_i^j – the number of occurrences of the value v_i^j for the confidential attribute S_j; in other words the ***descending ordered frequency set*** [11] for the confidential attribute S_j ($1 \leq j \leq r$ and $1 \leq i \leq s_j$). For each sensitive attribute S_j the following inequality holds: $f_1^j \geq f_2^j \geq \ldots \geq f_{s_j}^j$.
- SEC_i^j – the set of tuples from $\mathcal{M}$ such that they all have the value v_i^j for S_j ($1 \leq j \leq r$ and $1 \leq i \leq s_j$), in other words $SEC_i^j = \sigma_{S_j = v_i^j}(\mathcal{M})$. We use the term of ***a sensitive equivalence class*** or attribute S_j to refer to any SEC_i^j. The cardinality of SEC_i^j is f_i^j.
- cf_i^j – the ***cumulative descending ordered frequency set*** for the confidential attribute S_j ($1 \leq j \leq r$ and $1 \leq i \leq s_j$) [19]. In other words, $cf_i^j = \sum_{k=1}^{i} f_k^j$.
- $cf_i = \max_{j=1,r}(cf_i^j)$ $(0 \leq i \leq \min_{j=1,r}(s_j))$ – the ***maximum between ith cumulative descending ordered frequencies,*** for all sensitive attributes. We define $cf_0 = 0$.
- $pSEC_i^j = \begin{cases} SEC_i^j, & \text{if } i < p \\ SEC_p^j \cup SEC_{p+1}^j \cup \ldots \cup SEC_{s_j}^j, & \text{if } i = p \end{cases}$, ($1 \leq j \leq r$ and $1 \leq i \leq p$). We call each $pSEC_i^j$ as a ***p-sensitive equivalence class*** of attribute S_j. Each sensitive attribute S_j partitions the tuples in $\mathcal{M}$ in p p-sensitive equivalence classes. Moreover, the size of these equivalence classes descends from the $pSEC_1^j$ to $pSEC_{p-1}^j$. The last p-sensitive equivalence class, $pSEC_p^j$, does not follow this pattern.

P-sensitive *k*-anonymity can not be enforced for any given $\mathcal{IM}$, for any *p* and *k*. We present next two necessary conditions that express when this is possible [19].

Condition 1. (*First necessary condition for an $\mathcal{MM}$ to have p-sensitive k-anonymity property*): The minimum number of distinct values for each confidential attribute in $\mathcal{IM}$ must be greater than or equal to *p*.

A second necessary condition establishes the maximum possible number of *QI*-clusters in the masked microdata that satisfy *p*-sensitive *k*-anonymity. To specify this upper bound we use the maximum between cumulative descending ordered frequencies for each sensitive attribute in $\mathcal{IM}$ [19].

Condition 2. (*Second necessary condition for a $\mathcal{MM}$ to have p-sensitive k-anonymity property*): The maximum possible number of *QI*-clusters in the masked microdata is $\boldsymbol{maxClusters} = \min_{i=1,p} \left\lfloor \frac{n - cf_{p-i}}{i} \right\rfloor$.

Proof: We assume that for a given *IM*, *k* and *p*, the maximum possible number of *QI*-clusters in the masked microdata $maxClusters > \min_{i=1,p} \left\lfloor \frac{n - cf_{p-i}}{i} \right\rfloor$. Let *iVal* be the *i* value for which $\frac{n-cf_{p-i}}{i}$ is minimum. We have:

$$maxClusters > \frac{n - cf_{p-iVal}}{iVal} \text{ and } maxClusters \cdot iVal > n - cf_{p-iVal}. \tag{1}$$

Since cf_{p-iVal} tuples have only $p - iVal$ distinct values for a confidential attribute (from the definition of cumulative frequencies), the remaining tuples ($n - cf_{p-iVal}$) must contribute with at least *iVal* tuples to every cluster. In other words: $n-cf_{p-iVal} \geq maxClusters \cdot iVal$, relation that contradicts (1). Q.E.D.

Condition 2 provides a superior limit of the number of *p*-sensitive *QI*-clusters that can be formed in a microdata set, and not the actual number of such clusters that exist in data. Therefore, even the optimal partition w.r.t. the partition cardinality criterion could consist in less *p*-sensitive *QI*-clusters than the number estimated by Condition 2. Next, we give such an example where *maxClusters* value calculated according to Condition 2 is strictly greater than the maximum number of *p*-sensitive equivalence classes within the microdata. Fig. 1 contains a microdata described by 3 sensitive attributes together with the corresponding f_i^j and cf_i^j values.

A	**B**	**C**
1	a	α
1	b	β
2	a	β
2	b	α

	s_j	f_1^j	f_2^j
$j = 1$	A	2	2
$j = 2$	B	2	2
$j = 3$	C	2	2

cf_1^j	cf_2^j
2	4
2	4
2	4
cf_1	cf_2
2	4

Fig. 1. A microdata with corresponding frequency / cumulative frequency set values

For p=2, $maxClusters = \min_{i=1,p} \left\lfloor \frac{n - cf_{p-i}}{i} \right\rfloor = \left\lfloor \frac{4-2}{1} \right\rfloor = 2$. In fact, only one group that is 2-sensitive can be formed with these tuples!

2.3 Extended *p*-Sensitive *k*-Anonymity Model

The values of the attributes, in particular the categorical ones, are often organized according to some hierarchies. Although Samarati and Sweeney introduced the concept of value generalization hierarchy for only quasi-identifier attributes [16, 17], these hierarchies can be applied and used for sensitive attributes as well. For example, in medical datasets, the sensitive attribute *Illness* has values as specified by the *ICD9* codes (see Fig. 2) [8]. The data owner may want to protect not only the leaf values as in the *p*-sensitive *k*-anonymity model, but also values found at higher levels. For example, the information that a person has *cancer* (not a leaf value in this case) needs to be protected, regardless of the cancer type she has (*colon cancer*, *prostate cancer*, *breast cancer* are examples of leaf nodes in this hierarchy). If *p*-sensitive *k*-anonymity property is enforced for the released microdata, it is possible that for one *QI*-cluster all of the *Illness* attribute values to be descendants of the *cancer* node in the corresponding hierarchy, therefore leading to disclosure. To avoid such situations, the extended *p*-sensitive *k*-anonymity model was introduced [5].

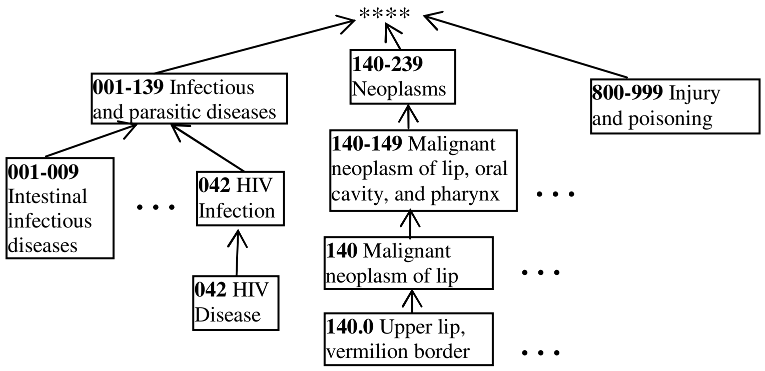

****
001-139 Infectious and parasitic diseases
140-239 Neoplasms
800-999 Injury and poisoning
001-009 Intestinal infectious diseases
042 HIV Infection
042 HIV Disease
140-149 Malignant neoplasm of lip, oral cavity, and pharynx
140 Malignant neoplasm of lip
140.0 Upper lip, vermilion border

Fig. 2. ICD9 disease hierarchy and codes

For the sensitive attribute S we use the notation HV_S to represent its value generalization hierarchy. We assume that the data owner has the following requirements in order to release a masked microdata:

- All ground values in HV_S must be protected against disclosure.
- Some non-ground values in HV_S must be protected against disclosure.
- All the descendants of a protected non-ground value in HV_S must also be protected.

Definition 4 (*strong value*): A protected value in the value generalization hierarchy HV_S of a confidential attribute S is called ***strong*** if none of its ascendants (including the root) is protected.

Definition 5 (*protected subtree*): We define a ***protected subtree*** of a hierarchy HV_S as a subtree in HV_S that has as root a strong protected value.

Definition 6 (*extended p-sensitive k-anonymity property*): The masked microdata ($\mathcal{MM}$) satisfies ***extended p-sensitive k-anonymity property*** if it satisfies *k*-anonymity and for each *QI*-cluster from $\mathcal{MM}$, and the values of each confidential attribute S within that group belong to at least p different protected subtrees in HV_S.

The necessary conditions to achieve extended *p*-sensitive *k*-anonymity on microdata are similar with the ones presented for *p*-sensitive *k*-anonymity model.

At a closer look, extended *p*-sensitive *k*-anonymity for a microdata is equivalent to *p*-sensitive *k*-anonymity for the same microdata where the confidential attributes values are generalized to their first protected ancestor starting from the hierarchy root (their strong ancestor). Consequently, in order to enforce extended *p*-sensitive *k*-anonymity to a dataset, the following two-steps procedure can be applied:

- Each value of a confidential attribute is generalized (temporarily) to its first strong ancestor (including itself).
- Any algorithm which can be used for *p*-sensitive *k*-anonymization is applied to the modified dataset. In the resulted masked microdata the original values of the confidential attributes are restored.

The dataset obtained following these steps respects the extended *p*-sensitive *k*-anonymity property.

3 Privacy Algorithms

Anonymization algorithms, besides achieving the properties required by the target privacy model (*p*-sensitive *k*-anonymity, *l*-diversity, (α, *k*)-anonymity, *t*-closeness), must also consider minimizing one or more cost measure. We know that optimal *k*-anonymization is a NP-hard problem [1]. By simple reduction to *k*-anonymity, it can be easily shown that *p*-sensitive *k*-anonymization is also a NP-hard problem. Several polynomial algorithms that achieve a suboptimal solution currently exist for enforcing *p*-sensitive *k*-anonymity and other similar models on microdata. In [6] we described a greedy clustering algorithm (*GreedyPKClustering*) for *p*-sensitive *k*-anonymity. For both *l*-diversity and (α, *k*)-anonymity the authors proposed to use adapted versions of Incognito as a first alternative [14, 22]. For (α, *k*)-anonymity a second algorithm based on local-recoding, called Top Down, was also presented [22]. Incognito and Top Down can be adapted for *p*-sensitive *k*-anonymity as well (in fact, we used such an adapted version of Incognito in our experiments for comparison purposes). The new anonymization algorithm will take advantage of the known properties of the *p*-sensitive *k*-anonymity model in order to improve the *p*-sensitive *k*-anonymous solutions w.r.t. various cost measures.

In the next two subsections we formally describe our approach to the anonymization problem, we present several cost measures, and we introduce our anonymization algorithm.

3.1 Problem Description

The microdata *p*-sensitive *k*-anonymization problem can be formulated as follows:

Definition 7 (*p-sensitive k-anonymization problem*): Given a microdata $\mathcal{IM}$, the ***p-sensitive k-anonymization problem*** for $\mathcal{IM}$ is to find a partition $\mathcal{S} = \{cl_1, cl_2, \ldots , cl_v\}$ of $\mathcal{IM}$, where $cl_j \subseteq \mathcal{IM}$, $j=1..v$, are called clusters and: $\bigcup_{j=1}^{v} cl_j = \mathcal{IM}$; $cl_i \bigcap cl_j = \emptyset$, $i, j = 1..v$, $i \neq j$; $|cl_j| \geq k$ and cl_j is *p*-sensitive, $j=1..v$; and a cost measure is optimized.

Once a solution $\mathcal{S}$ to the above problem is found for a microdata $\mathcal{IM}$, a masked microdata $\mathcal{MM}$ that is *p*-sensitive and *k*-anonymous is formed by generalizing the quasi-identifier attributes of all tuples inside each cluster of $\mathcal{S}$ to the same values. The generalization method consists in replacing the actual value of an attribute with a less specific, more general value that is faithful to the original [17].

For categorical attributes we use generalization based on predefined hierarchies [9]. For numerical attributes we use the hierarchy-free generalization [12], which consists in replacing the set of values to be generalized to the smallest interval that includes all the initial values. For instance, the values: 25, 39, 36 are generalized to the interval [25-39]. It is worth noting that the values for sensitive attributes remain unchanged within each cluster.

The anonymization of the initial microdata must be conducted to preserve data usefulness and to minimize information loss. In order to achieve this goal, we generalize each cluster to the least general tuple that represents all tuples in that group. We call *generalization information* for a cluster the minimal covering tuple for that cluster, and we define it as follows.

Definition 8 (*generalization information*): Let $cl = \{r_1, r_2, \ldots, r_q\} \in \mathcal{S}$ be a cluster, $\mathcal{KN} = \{N_1, N_2, ..., N_s\}$ be the set of numerical quasi-identifier attributes and $\mathcal{KC} = \{C_1, C_2,, \ldots, C_t\}$ be the set of categorical quasi-identifier attributes. The ***generalization information of cl***, w.r.t. quasi-identifier attribute set $\mathcal{K} = \mathcal{KN} \cup \mathcal{KC}$ is the "tuple" $gen(cl)$, having the scheme $\mathcal{K}$, where:

- For each categorical attribute $C_j \in \mathcal{K}$, $gen(cl)[C_j]$ = the lowest common ancestor in $\mathcal{H}_{Cj}$ of $\{r_1[C_j], r_2[C_j], \ldots, r_q[C_j]\}$, where $\mathcal{H}_C$ denotes the hierarchies (domain and value) associated to the categorical quasi-identifier attribute C;
- For each numerical attribute $N_j \in \mathcal{K}$, $gen(cl)[N_j]$ = the interval $[\min\{r_1[N_j], r_2[N_j], \ldots, r_q[N_j]\}, \max\{r_1[N_j], r_2[N_j], \ldots, r_q[N_j]\}]$.

For a cluster cl, its generalization information $gen(cl)$ is the tuple having as value for each quasi-identifier attribute, numerical or categorical, the most specific common generalized value for all that attribute values from cl tuples. In $\mathcal{MM}$, each tuple from the cluster cl will be replaced by $gen(cl)$.

There are several possible cost measures that can be used as optimization criterion for the p-sensitive k-anonymization problem [3, 4, etc.]. A simple cost measure is based on the size of each cluster from $\mathcal{S}$. This measure, called *discernability metric* (*DM*) [3] assigns to each record x from $\mathcal{IM}$ a penalty that is determined by the size of the cluster containing x:

$$DM(\mathcal{S}) = \sum_{j=1}^{v} (|cl_j|)^2 . \tag{2}$$

LeFevre introduced an alternative measure, called the *normalized average cluster size metric* (*AVG*) [12]:

$$AVG(\mathcal{S}) = \frac{n}{v \cdot k} , \tag{3}$$

where n is the size of the $\mathcal{IM}$, v is the number of clusters, and k is as in k-anonymity.

It is easy to notice that the *AVG* cost measure is inversely proportional with the number of clusters, and minimizing *AVG* is equivalent to maximizing the total number of clusters.

The last cost measure we present is the information loss caused by generalizing each cluster to a common tuple [4, 20]. This is an obvious measure to guide the partitioning process, since the produced partition $\mathcal{S}$ will subsequently be subject to cluster-level generalization.

Definition 9 (***cluster information loss***)**:** Let *cl* ∈ $\mathcal{S}$ be a cluster, *gen*(*cl*) its generalization information and $\mathcal{K} = \{N_1, N_2, .., N_s, C_1, C_2, .., C_t\}$ the set of quasi-identifier attributes. The ***cluster information loss*** caused by generalizing *cl* tuples to *gen*(*cl*) is:

$$IL(cl) = |cl| \cdot \left(\sum_{j=1}^{s} \frac{size(gen(cl)[N_j])}{size\left(\left[\min_{r \in \mathcal{IM}} r[N_j],\ \max_{r \in \mathcal{IM}} r[N_j]\right]\right)} + \sum_{j=1}^{t} \frac{height(\Lambda(gen(cl)[C_j]))}{height(H_{C_j})} \right), \tag{4}$$

where:

- $|cl|$ denotes the cluster *cl* cardinality;
- $size([i_1, i_2])$ is the size of the interval $[i_1, i_2]$ (the value i_2-i_1);
- $\Lambda(w)$, $w \in H_{Cj}$ is the subhierarchy of H_{Cj} rooted in w;
- $height(H_{Cj})$ denotes the height of the tree hierarchy H_{Cj}.

Definition 10 (***total information loss***)**:** ***Total information loss for*** a solution $\mathcal{S} = \{cl_1, cl_2, \ldots, cl_v\}$ of the *p*-sensitive *k*-anonymization problem, denoted by $IL(\mathcal{S})$, is the sum of the information loss measure for all the clusters in $\mathcal{S}$:

$$IL(\mathcal{S}) = \sum_{j=1}^{v} (IL(cl_j))\ . \tag{5}$$

In order to achieve *p*-sensitive *k*-anonymity for each cluster, we need to address the *p*-sensitive part with uttermost attention. While the *k*-anonymity is satisfied for each individual cluster when its size is *k* or more, the *p*-sensitive property is not so obvious to achieve. To help us in this process we introduce two diversity measures that quantify, with respect to sensitive attributes, the diversity between a tuple and a cluster and the homogeneity of a cluster.

Let X^i, $i = 1 \ldots n$, be the tuples from $\mathcal{IM}$ subject to *p*-sensitive *k*-anonymization. We denote an individual tuple by $X^i = \{k_1^i, \ldots, k_q^i, s_1^i, \ldots, s_r^i\}$, where k^i s are the values for the quasi-identifier attributes and s^i s are the values for the confidential attributes.

Definition 11 (***diversity between a tuple and a cluster***)**:** The ***diversity between a tuple X^i and a cluster cl*** w.r.t. the confidential attributes is given by:

$$Div(X^i, cl) = \sum_{i=1}^{r} (y_i^{'} - y_i) \cdot (p - y_i) \cdot w_i \text{, where} \tag{6}$$

- y_i – is the number of distinct values for attribute S_i ($1 \le i \le r$) in *cl* if this number is less than *p*, and *p* otherwise.

- y_i' – is the number of distinct values for attribute S_i ($1 \le i \le r$) in $cl' = cl \cup \{X^i\}$ if this number is less than p, and p otherwise. It is easy to show that for each i = 1 … r, y_i' is either y_i or $y_i + 1$.
- (w_1, w_2, …, w_r) – is a weight vector, $\sum_{l=1}^{r} w_l = 1$. The data owner can choose different criteria to define this weights vector. One possible selection of the weight values is to initialize them as inversely proportional to the number of distinct sensitive attribute values in the microdata $\mathcal{IM}$ (defined as s_i values). In the experimental section we chose to use the same value for all the weights.

Definition 12 (*cluster homogeneity*): The ***homogeneity of a cluster cl*** w.r.t. the confidential attributes is given by:

$$Hom(cl) = \sum_{i=1}^{r} (p - y_i) \cdot w_i\,, \tag{7}$$

where y_i and w_i have the same meaning as in the previous definition.

Property 1: A cluster *cl* is *p*-sensitive w.r.t. all confidential attributes S_1, S_2, …, S_r iif $Hom(cl)=0$.

Proof: This property follows directly from the definition of cluster homogeneity.

3.2 The *EnhancedPKClustering* Algorithm

First, we introduce two total order relations that will help us present our algorithm.

Definition 13 (*$\ge_h$ relation*): Let S_i and S_j be two sensitive attributes. The following relation $S_i \ge_h S_j$ is true if and only if $maxClusters_i \le maxClusters_j$ where $maxClusters_l$ is computed for $\mathcal{IM}$ with only one sensitive attribute S_l, $l = i, j$, given *p* and *k*. We use the term ***S_i is harder than or as hard as S_j to make sensitive*** for $S_i \ge_h S_j$.

Definition 14 (*$\ge_d$ relation*): Let cl_i and cl_j be two clusters. The following relation $cl_i \ge_d cl_j$ is true if and only if $Hom(cl_i) \le Hom(cl_j)$, for a given *p*. We use the term ***cl_i is more diverse than or as diverse as cl_j*** for $cl_i \ge_d cl_j$.

Property 2: Let *maxClusters* be as defined in Section 2.2. Let S_1 harder than or as hard as every other confidential attribute to make sensitive as defined in Definition 13. Let *iVal* be the smallest value between 1 and *p* such that $maxClusters = \left\lfloor \frac{n - cf_{p-iVal}}{iVal} \right\rfloor$. The relation $|SEC_i^1| \le maxClusters$ holds for all $i \ge p-iVal+1$ for which SEC_i^1 are defined.

Proof: From the definition of sensitive equivalence classes, the larger the value of *i* the smaller the cardinality of *SEC*'s; therefore, it is enough to prove that $|SEC_{p-iVal+1}^1| \le maxClusters$ holds.

From *maxClusters* definition and the selection of *iVal* we have:

$$maxClusters = \left\lfloor \frac{n - cf_{p-iVal}}{iVal} \right\rfloor < \left\lfloor \frac{n - cf_{p-iVal+1}}{iVal - 1} \right\rfloor \tag{8}$$

As S_1 is the hardest to make sensitive attribute and from definition of cumulative frequencies it follows that:

$$cf_{p-iVal+1} \geq cf^1_{p-iVal+1} = cf^1_{p-iVal} + | SEC^1_{p-iVal+1} | = cf_{p-iVal} + | SEC^1_{p-iVal+1} | \tag{9}$$

From (8) and (9) the following relation holds:

$$\left\lfloor \frac{n - cf_{p-iVal}}{iVal} \right\rfloor < \left\lfloor \frac{n - (cf_{p-iVal} + | SEC^1_{p-iVal+1} |)}{iVal - 1} \right\rfloor \tag{10}$$

Assuming $| SEC^1_{p-iVal+1} | > maxClusters$ $\Rightarrow$

$$| SEC^1_{p-iVal+1} | > \frac{n - cf_{p-iVal}}{iVal} \tag{11}$$

Using relations (10) and (11) we obtain**:**

$$\left\lfloor \frac{n - cf_{p-iVal}}{iVal} \right\rfloor < \left\lfloor \left(n - \left(cf_{p-iVal} + \frac{n - cf_{p-iVal}}{iVal} \right) \right) \Big/ (iVal - 1) \right\rfloor = \left\lfloor \frac{n - cf_{p-iVal}}{iVal} \right\rfloor. \tag{12}$$

As a result, our assumption is false and the property $| SEC^1_i | \leq maxClusters$ holds for all $i \geq p\text{-}iVal+1$. Q.E.D.

The *EnhancedPKClustering* algorithm finds a solution for the *p*-sensitive *k*-anonymization problem for a given *IM*. It considers *AVG* (or the partition cardinality) that has to be maximized as the cost measure.

This algorithm starts by enforcing the *p*-sensitive part using the properties proved for the *p*-sensitive *k*-anonymity model. The tuples from *IM* are distributed to form *p*-sensitive clusters with respect to the sensitive attributes. After *p*-sensitivity is achieved, the clusters are further processed to satisfy k-anonymity requirement as well. A more detailed description of how the algorithm proceeds follows.

In the beginning, the algorithm determines the *p*-sensitive equivalence classes, orders the attributes based on the harder to make sensitive relation, and computes the value *iValue* that divides the *p*-sensitive equivalence classes into two categories: one with less frequent values for the hardest to anonymize attribute and one with more frequent values. Now, the *QI*-clusters are created using the following steps:

- First, the tuples in the less frequent category of *p*-sensitive equivalence classes are divided into *maxClusters* clusters (*Split* function) such that each cluster will have *iValue* tuples with *iValue* distinct values within each cluster for attribute S_1 (the hardest to anonymize).
- Second, the remaining *p*-sensitive equivalence classes are used to fill the clusters such that each of them will have exactly *p* tuples with *p* distinct values for S_1.
- Third, the tuples not yet assigned to any cluster are used to add diversity for all remaining sensitive attributes until all clusters are *p*-sensitive. If no tuples are available, some of the less diverse (more homogenous) clusters are removed and their tuples are reused for the remaining clusters. At the end of this step all clusters are *p*-sensitive.

- Fourth, the tuples not yet assigned to any cluster are used to increase the size of each cluster to *k*. If no tuples are available, some of the less populated clusters are remov`ed and their tuples are reused for the remaining clusters. At the end of this step all clusters are *p*-sensitive *k*-anonymous.

Along all the steps, when a choice is to be made, one or more optimization criteria are used (diversity between a tuple and a cluster, and increase in information loss).

```
Algorithm EnhancedPKClustering is
Input  $\mathcal{IM}$ - initial microdata;
       $p$, $k$ - as in $p$-sensitive $k$-anonymity;
Output $\mathcal{S} = \{cl_1, cl_2, \ldots, cl_v\}$ - a solution for the $p$-sensitive $k$-anonymi-
       zation problem for $\mathcal{IM}$;

Reorder $S_1$, $S_2$, …, $S_r$ such that $S_i \geq_h S_j$, $i$, $j$ = 1..$v$, $i > j$;
$maxClusters = \min_{i=1,p} \left\lfloor \frac{n - cf_{p-i}}{i} \right\rfloor$;
$iValue = \min \left\{ i \mid maxClusters = \left\lfloor \frac{n - cf_{p-i}}{i} \right\rfloor, i = 1..p \right\}$;
for $i$ = 1 to $maxClusters$ do $cl_i = \emptyset$;
$S = \{cl_1, cl_2, \ldots, cl_{maxClusters}\}$;
$U = \{pSEC^1_{p-iValue+1}, pSEC^1_{p-iValue+2}, \ldots, pSEC^1_p\}$;
// Based on Condition 2, the tuples in $U$ can be allocated to
// $maxClusters$ clusters, each having $iValue$ different values for $S_1$
Split ($U$, $\mathcal{S}$, $E$);
for $j$ = $p-iValue$ down to 1 {
  auxSEC = $pSEC^1_j$; aux$\mathcal{S}$ = $\mathcal{S}$;
  while (aux$\mathcal{S} \neq \emptyset$) {
    ($tuple$, $cl$) = BestMatch($auxSEC$, $aux\mathcal{S}$);  // maximize diversity
    cl = cl $\cup$ {tuple};
    auxSEC = auxSEC - {tuple};
    aux$\mathcal{S}$ = aux$\mathcal{S}$ - {cl};
  }  // end while
}  // end for.
// Now $p$-sensitive property holds w.r.t. $S_1$

// $T$ contains leftover tuples from $pSEC$'s plus tuples from $E$.
Let $T$ be the set of tuples not assigned yet to any cluster from $S$.
Reorder clusters from $S$, such that $cl_i \geq_d cl_j$, $i,j$ = 1..$maxClusters$, $i>j$;
h = 1;
while (Hom($cl_h$) == 0) h= h + 1;
//$cl_h$ the first cluster without p-sensitivity
aux = $maxClusters$;
while (h $\leq$ aux) {
  while (h $\leq$ aux) && (T $\neq \emptyset$) {
    ($tuple$, $cl_h$) = BestMatch($T$, {$cl_h$});
    $cl_h$ = $cl_h$ $\cup$ {$tuple$}; $T$ = $T$ - {$tuple$};
    if (Hom($cl_h$) == 0) h = h + 1;
  }
  if ($T$ == $\emptyset$) && ($h \leq aux$) {
    $T$ = $cl_{aux}$;
    $aux$ = $aux$ - 1; // redistribute T
  }
} // p-sensitivity property holds for all clusters.
```

```
// the set T (possible empty) must be spread.
Reorder $\mathcal{S}$ based on the number of tuples in each cluster($|cl_i| \geq |cl_j|$,
$i,j = 1..aux$, $i > j$;)
$u = 1$;
while ($|cl_u| \geq k$) $u = u + 1$;
// $cl_i$ with $i > u$ are not $k$-anonymous.
v = $\min\left(\text{aux}, \text{u} + \left\lfloor \frac{|T| + |cl_{u+1}| + ... + |cl_{aux}|}{k} \right\rfloor\right)$;
if ($v$ < aux) $T = T \cup \{t \in cl_i \mid i = v + 1, .., aux\}$;
for $i$ = 1 to $totalClusters$ do {
  while ($|cl_i| < k$) {
    Find a $tuple$ such that IL($cl_i \cup \{tuple\}$) = min{IL($cl_i \cup \{t\}$) | $t \in T$);
    $cl_i = cl_i \cup \{tuple\}$;
    $T = T - \{tuple\}$;
  }
} // $p$-sensitive $k$-anonymity is achieved

for $every\ t \in T$ do {    // extra tuples left in $T$ are distributed
  Find $cl$ such that IL($cl \cup \{t\}$) - IL($cl$)
     = min(IL($cl_i \cup \{t\}$) - IL($cl_i$) | $i = 1,..,v$);
  $cl = cl \cup \{t\}$;
}
End **EnhancedPKClustering**;

Function **Split**($U$, $\mathcal{S}$, $E$)
  $U = \{pSEC^1_{p-iValue+1},.., pSEC^1_p\} = \{SEC^1_{p-iValue+1}, .., SEC^1_p, SEC^1_{p+1}, .., SEC^1_{s_1}\}$;
  $i = 1$;
  for $j = s_1$ down to $p - iValue + 1$ do {
    $auxSEC = SEC^1_j$;
    // tuples are assigned to clusters in a circular way; any two tuples
    // from the same $auxSEC$ will belong to distinct clusters. (Prop. 2)
    while ($auxSEC \neq \emptyset$) {
      ($t$, $cl_i$) = BestMatch($auxSEC$, $\{cl_i\}$);
      $auxSEC = auxSEC - \{t\}$;
      $cl_i = cl_i \cup \{t\}$;
      $i = i + 1$;
      if ($i > |\mathcal{S}|$) then
        if ($|cl_1| < iValue$) then $i = 1$
        else {
           // each cluster has $iValue$ tuples
           E = all tuples in U not assigned; return; }
    }
  }
End **Split**;

Function **BestMatch**($auxSEC$, $aux\mathcal{S}$)
  Find the set $Pairs$ of all pairs ($t_i$, $cl_j$) such that Div($t_i$,$cl_j$) =
    max{Div($t$,$cl$) | ($t$,$cl$) $\in auxSEC \times aux\mathcal{S}$};   // maximize diversity
  Return any pair ($t$,$cl$) $\in Pairs$ such that IL($cl \cup \{t\}$)-IL($cl$) =
    min{IL($cl_j \cup \{t_i\}$)-IL($cl_j$) | ($t_i$,$cl_j$) $\in Pairs$};  // minimize $IL$
End **BestMatch**;
```

Informally, we state that the complexity of the *EnhancedPKClustering* algorithm is O(n^2). A complete complexity analysis of the algorithm will be presented in the full version of the paper.

4 Preliminary Results

In this section we report the experiments we have conducted to compare, for the *p*-sensitive *k*-anonymity model, the performance of *EnhancedPKClustering* algorithm against: an adapted version of *Incognito* algorithm [11] and the *GreedyPKClustering* algorithm [6]. We intend to extend our experiments and perform comparative tests with other algorithms proposed to enforce models equivalent with *p*-sensitive *k*-anonymity (*l*-diversity, (α, *k*)-anonymity, and *t*-closeness). However, we think that an algorithm based on global recoding will produce weaker results (in terms of any cost measure) compared to a local recoding algorithm (such as *EnhancedPKClustering* or *GreedyPKClustering*), and this without connection to a specific anonymity model.

All three algorithms have been implemented in Java, and tests were executed on a dual CPU machine running Windows 2003 Server with 3.00 GHz and 1 GB of RAM.

A set of experiments has been conducted for an *IM* consisting in 10000 tuples randomly selected from the *Adult* dataset from the UC Irvine Machine Learning Repository [15]. In all the experiments, we considered *age, workclass, marital-status, race*, *sex*, and *native-country* as the set of quasi-identifier attributes; and *education_num*, *education*, and *occupation* as the set of confidential attributes. Microdata *p*-sensitive *k*-anonymity was enforced in respect to the quasi-identifier consisting of all 6 quasi-identifier attributes and all 3 confidential attributes. Although many values of *k* and *p* were considered, due to space limitations, we present in this paper only a small subset of the results.

Fig. 3 shows comparatively the AVG and DM values of the three algorithms, *EnhancedPKClustering*, *GreedyPKClustering* and *Incognito*, produced for *k* = 20 and different *p* values. As expected, the results for the first two algorithms clearly outperform Incognito results. We notice that *EnhancedPKClustering* is able to improve the performances of the *GreedyPKClustering* algorithm in cases where solving the *p*-sensitivity part takes prevalence over creating clusters of size *k*.

Fig. 4 left shows comparatively the DM and AVG values obtained by *EnhancedPKClustering* algorithm divided by the same values computed using *GreedyPKClustering* algorithm. We notice that for *p* = 2 and 4 there is no improvement. In these cases both algorithms were able to find the optimal solution in terms of DM and AVG values. As soon as the *p*-sensitive part is hard to achieve, the *EnhancedPKClustering* algorithm performs better. Fig. 4 right shows the time required to generate the masked microdata by all three algorithms. Since *Incognito* uses global recording and our domain generalization hierarchies for this dataset have a low height, the running time is very fast. The *GreedyPKClustering* is faster than the new algorithm for small values of p, but when it is mode difficult to create p-sensitivity within each cluster the *EnhancedPKClustering* has a slight advantage. Based on these results, it is worth noting that a combination of GreedyPKClustering

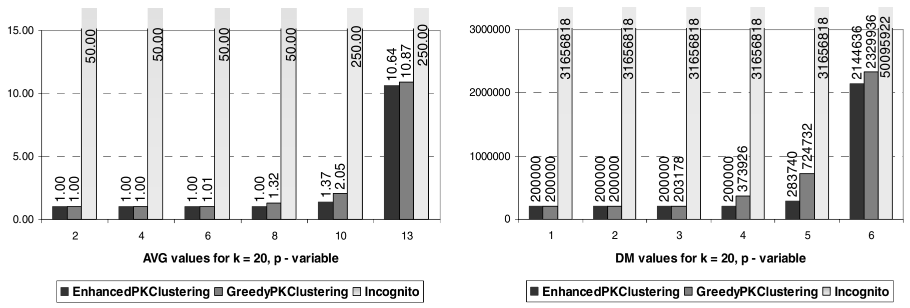


Fig. 3. AVG and DM for *EnhancedPKClustering*, *GreedyPKClustering*, and *Incognito*

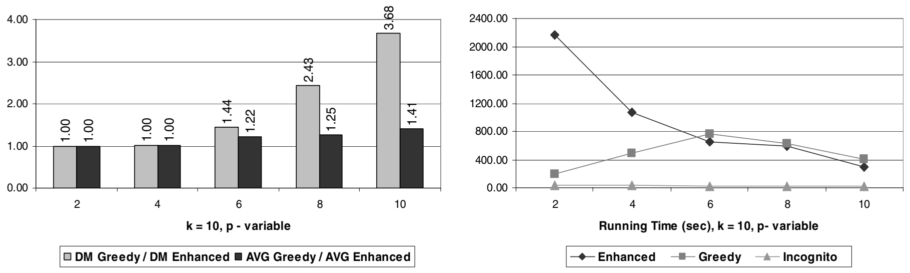


Fig. 4. Comparison between *EnhancedPKClustering* and *GreedyPKClustering* in terms DM and AVG values and the running time of all three algorithms

(for low values of *p*, in our experiment 2 and 4) and *EnhancedPKClustering* (for high values of *p*, in our experiment 6, 8, and 10) would be desirable in order to improve both running time and the selected cost measure (AVG or DM).

5 Conclusions and Future Work

In this paper, a new algorithm to generate masked microdata with *p*-sensitive *k*-anonymity property was introduced. The algorithm uses several properties of the *p*-sensitive *k*-anonymity model in order to efficiently create the masked microdata that satisfy the privacy requirement. Our experiments have shown that the proposed algorithm improves both AVG and DM cost measures over existing algorithms. As our algorithm is based on local recoding (cluster-level generalization) and accepts multiple sensitive attributes, it leads to better results than the *Incognito* algorithm, but it also outperforms the local recoding based *GreedyPKClustering* algorithm. Two diversity measures that help characterize this similarity of sensitive attributes values within each cluster are also introduced.

We believe that the *EnhancedPKClustering* algorithm could be used for enforcing (α, k)-anonymity, *l*-diversity, or the new introduced *t*-closeness on microdata as well.

Acknowledgments. This work was supported by the Kentucky NSF EPSCoR Program under grant "*p*-Sensitive *k*-Anonymity Property for Microdata".

References

1. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Anonymizing Tables. In: Proceedings of the ICDT, pp. 246–258 (2005)
2. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic Databases. In: Proceedings of the VLDB, pp. 143–154 (2002)
3. Bayardo, R.J, Agrawal, R.: Data Privacy through Optimal k-Anonymization. In: Proceedings of the IEEE ICDE, pp. 217–228. IEEE Computer Society Press, Los Alamitos (2005)
4. Byun, J.W., Kamra, A., Bertino, E., Li, N.: Efficient k-Anonymity using Clustering Technique. CERIAS Tech. Report 2006-10 (2006)
5. Campan, A., Truta, T.M.: Extended P-Sensitive K-Anonymity. Studia Universitatis Babes-Bolyai Informatica 51(2), 19–30 (2006)
6. Campan, A., Truta, T.M., Miller, J., Sinca, R.A.: Clustering Approach for Achieving Data Privacy. In: Proceedings of the International Data Mining Conference (2007)
7. HIPAA.: Health Insurance Portability and Accountability Act (2002), Available online at: http://www.hhs.gov/ocr/hipaa
8. ICD9.: International Classification of Diseases. Available online at: http://icd9cm.chrisendres.com/index.php
9. Iyengar, V.: Transforming Data to Satisfy Privacy Constraints. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 279–288. ACM Press, New York (2002)
10. Lambert, D.: Measures of Disclosure Risk and Harm. Journal of Official Statistics 9, 313–331 (1993)
11. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Incognito: Efficient Full-Domain K-Anonymity. In: Proceedings of the ACM SIGMOD, pp. 49–60. ACM Press, New York (2005)
12. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Mondrian Multidimensional K-Anonymity. Proceedings of the IEEE ICDE, 25 (2006)
13. Li, N., Li, T., Venkatasubramanian, S.: T-Closeness: Privacy Beyond k-Anonymity and l-Diversity. Proceedings of the IEEE ICDE (2007)
14. Machanavajjhala, A., Gehrke, J., Kifer, D.: L-Diversity: Privacy beyond K-Anonymity. Proceedings of the IEEE ICDE, 24 (2006)
15. 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
16. Samarati, P.: Protecting Respondents Identities in Microdata Release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
17. Sweeney, L.: k-Anonymity: A Model for Protecting Privacy. and Knowledge-based Systems 10(5), 557–570 (2002)
18. 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)
19. Truta, T.M., Bindu, V.: Privacy Protection: P-Sensitive K-Anonymity Property. Proceedings of the Workshop on Privacy Data Management, In Conjunction with IEEE ICDE, 94 (2006)
20. Truta, T.M., Campan, A.: K-Anonymization Incremental Maintenance and Optimization Techniques. In: Proceedings of the ACM SAC, pp. 380–387. ACM Press, New York (2007)

21. Winkler, W.: Matching and Record Linkage. In: Business Survey Methods, Wiley, Chichester (1995)
22. 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 ACM KDD, pp. 754–759. ACM Press, New York (2006)
23. Xiao, X., Tao, Y.: Personalized Privacy Preservation. In: Proceedings of the ACM SIGMOD, pp. 229–240. ACM Press, New York (2006)

Preventing Privacy-Invasive Software Using Collaborative Reputation Systems

Martin Boldt, Bengt Carlsson, Tobias Larsson, and Niklas Lindén

Blekinge Institute of Technology, Box 520, SE-372 25, Sweden
{martin.boldt,bengt.carlsson,tola01,nili02}@bth.se

Abstract. Privacy-invasive software, loosely labelled spyware, is an increasingly common problem for today's computer users, one to which there is no absolute cure. Most privacy-invasive software is positioned in a legal grey zone, as the user accepts the malicious behaviour when agreeing to the End User License Agreement. This paper proposes the use of a specialized reputation system to gather and share information regarding software behaviour between community users. A client application helps guide the user at the point of executing software on the local computer, displaying other users' feedback about the expected behaviour of the software. We discuss important aspects to consider when constructing such a system, and propose possible solutions. Based on the observations made, we implemented a client/server based proof-of-concept tool, which allowed us to demonstrate how such a system would work. We also compare this solution to other, more conventional, protection methods such as anti-virus and anti-spyware software.

Keywords: Information security, malware prevention, reputation systems.

1 Introduction

Our society is continuously moving in an increasingly more computerized direction where software has a central role [15][28]. Because of this development computer users are in need of more aiding mechanisms to help them distinguishing legitimate software from its questionable counterparts. Without such mechanisms we will experience a gradual increase in the negative consequences resulted by such software, affecting more and more of our daily lives by involving for instance mobile devices and media centres. Sources indicate that well over 80% of all home PCs and more than 30% of all corporate PCs connected to the Internet are infected by questionable software, often labelled *spyware* [32][37]. Affected computer owners are not aware of the fact that their computer is infected with spyware since they rely entirely on anti-virus software and firewalls to protect them. However, anti-virus software does not focus on spyware, but rather on more malicious software types, such as viruses, worms and Trojan horses [2].

Although some spyware programs might be malicious, many are considered to be legitimate software distributed by highly profitable companies that are gathering information about its users, showing targeted ads, sending user behaviour patterns, visited websites and similar, storing them for an unknown period of time as user profiles in central databases. Spyware are often in a legal grey zone since they normally inform the users of their actions, but often in such a format that it is unrealistic to believe that normal computer users will read and understand the provided information. The *End User License Agreement* (EULA) that the user has to agree on before using or installing the software are often written in a legal format, sometimes spanning well over 5000 words, and most users choose to proceed without actually studying it, giving his or her consent to whatever might be stated in the EULA, i.e. anything the software developer wants [7][14][31].

There are numerous ongoing projects and attempts to produce effective countermeasures for removing spyware [22][25][33]. However, this requires a classification of some software as “harmful to the user” which is legally problematic. The main reason for this is because the information regarding system behaviour is stated in the license agreement that the user already has accepted, which could lead to law suits [14][34]. Such legal disputes have already proved to be costly for anti-spyware software companies [29]. As a result of this, they may be forced to remove certain software from their list of targeted spyware to avoid future legal actions, and hence deliver an incomplete product to their customers, being unable to correctly classify some software as privacy-invasive.

As the problem of spyware is widely spread, and no complete protection is available, there is a need for ways to better inform the user about the software he or she uses, while still not classifying it as “harmful to the user” and hence risking law suits. There are numerous well-known and popular websites based on the concept of letting users grade different services, applications, shops, and similar e.g., Flixster, IMDb.com, and Pricerunner [12][18][26]. The main concept is to help guide other consumers to, for example, find the best store to shop at and to avoid pitfalls and unethical sellers [38]. We have combined this concept with a client software that helps guide the user whenever a program is about to execute on his computer, by showing other users rating and comments of the particular software. Larsson and Lindén implemented this idea into a proof-of-concept tool during their masters thesis work[1][21]. In this system the users are asked to rate their most frequently used software, by grading it between 1 and 10. In return they are given access to aggregated ratings for all software in the reputation system. By using the knowledge from previous users it is possible for new users to reach more informed decisions when installing a specific software, i.e. allowing them to stop questionable software *before* it enters their computer. The proof-of-concept tool has found a group of continuous users, which has rendered in well over 2000 rated software programs in the reputation database.

1.1 Background and Related Work

The usage of the term spyware has become increasingly popular, both by users, media and software vendors [1][30]. It has been defined as software that “track users’ activities online and offline, provide targeted advertising, and/or engage in other types of activities that users describe as invasive or undesirable [8][13]. This means that it

[1] The tool is available for free from: http://www.softwareputation.com

has come to include all kinds of malicious software, ranging from software that displays advertisements based on user behaviour (adware) to Trojan key loggers, as well as actual spying software (spyware) [31]. A better term to use instead of spyware, would be privacy-invasive software (PIS). In an attempt to clarify the usage of this term, Boldt and Carlsson based their classification of privacy-invasive software on user's informed consent and negative user consequence, as shown in Table 1 [4][5].

Table 1. Classification of privacy-invasive software with respect to user's informed consent (high, medium and low) and negative user consequences (tolerable, moderate and severe)

	Tolerable Negative Consequences	**Moderate Negative Consequences**	**Severe Negative Consequences**
High Consent	1) Legitimate software	2) Adverse software	3) Double agents
Medium Consent	4) Semi-transparent software	5) Unsolicited software	6) Semi-parasites
Low Consent	7) Covert software	8) Trojans	9) Parasites

User consent is specified as either *low*, *medium* or *high*, while the degree of negative consequences span between *tolerable*, *moderate*, and *severe.* This classification allows us to first make a distinction between legitimate software and spyware, and secondly between spyware and malicious software (malware). All software that has low user consent, *or* which impairs severe negative consequences should be regarded as malicious software. While, on the other hand, any software that has high user consent, *and* which results in tolerable negative consequences should be regarded as legitimate software. By this follows that spyware constitutes the remaining group of software, i.e. those that have medium user consent or which impair moderate negative consequences.

We base our work on Simone Fischer-Hübner's definition of *privacy*, in which she divides the concept into the following three areas [11]:

- *territorial privacy* focusing on the protection of the public area surrounding a person, such as the workplace or the public space
- *privacy of the person* which protect the individual from undue interference that constitute for instance physical searches and drug tests
- *informational privacy* protecting if and how personal information (information related to an identifiable person) is being gathered, stored, processed, and further disseminated.

Since our work has its origin in a computer setting we interpret the above three areas into a computer context. We argue that this is motivated since computers are being increasingly more weaved together with our daily lives, affecting individuals' privacy. Our classification of privacy-invasive software is related to the last two areas listed above, i.e. protecting the user from undue interference, and safeguarding users' personal information, while using computers. Therefore our view of privacy does not only focus on the communication of personal information, but it also includes undue interference that negatively affects the users' computer experience.

In an attempt to mitigate the negative effects from PIS we propose the use of a reputation system where computer users collaborate with the goal to distinguish legitimate software from PIS. As described by Resnick et al. a reputation system "collects, distributes, and aggregates feedback about participants' past behaviour" [27]. This can either be part of a larger system, to give the users incentives to behave well in the future knowing that other users will be able to review past transactions e.g. on an auction site, or as a system itself used for rating e.g. resellers of home appliances, Hollywood blockbusters, or basically any kind of product or service. This helps new users to establish a trust relationship towards a particular reseller or company based on other users' past opinions about the other party, without any personal contact with the reseller or company in question. This is increasingly important considering the present development rate for e-commerce and online services where customers seldom, if ever, meet the business representatives they are dealing with.

2 Important Considerations

There are two main issues that need to be addressed when considering the design and implementation of the proposed system. How to protect users' privacy and at the same time address incorrect information in the system. We will address these two considerations individually; explaining the problem at hand, as well as proposing one or more possible solutions that may help prevent the problem, or at least reduce the impact of it [9].

2.1 Addressing Incorrect Information

There are a number of aspects to take into consideration when building a system that is to gather, store and present information from multiple, unknown users. Although the system has been set up for a clear purpose, individual users, or groups of users, may find it more interesting to – for instance – intentionally enter misleading information to discredit a software vendor they dislike, use multiple computers in a distributed attack against the system to fill the database with bogus votes, enter irrelevant or indecent comments, and so on. When it comes to inventing new ways of disturbing peace, the stream of ideas seems to be never-ending.

Even though it may be done without malice, even in good faith, ignorant users voting and leaving feedback on programs they know nothing or little about may be a rather big problem for a software reputation system, especially at a budding phase. If the number of users is low, compared to the number of software to be rated, there is a big risk that many software will be without any, or with just a few, votes. Even worse, if these few votes and comments have been given by users with little actual knowledge about the software they are rating, they may – for example – give the installer of a program bundled with many different PIS a high rating, commenting that it is a great free and highly recommended program. In a normal environment, this would not be a problem, as a number of more experienced users would already have added negative feedback, warning other users of the potential dangers with installing this software package. However, in the cases where there are few users and votes available at any point of time, this may be a big problem.

We have identified three different approaches to mitigate the problem with unintentionally incorrect information. The first one involves allowing the users to rate not only the software but also the feedback of other users in terms of helpfulness, trustworthiness and correctness, creating a reliability profile for each user. This profile could be thought of as a trust factor that is used to weight the ratings of different users, making the votes and comments of well-known, reliable users more visible and influential than those of new users. It does not directly handle the problem of inexperienced users giving incorrect information and ratings, if they are the only ones commenting and voting, but as soon as more experienced users give contradicting votes, their opinions will carry a higher weight, tipping the balance in a – hopefully – more correct direction.

The second approach is to use bootstrapping of the program database at an early stage, preferably before the system is put to use, copying the information from an existing, more or less reliable, software rating database of programs and their individual ratings into the database of the reputation system. That way, it would be possible to ensure that no common program has few or zero votes, and in the event of novice users giving the software unfair positive or negative ratings and comments, the number of existing votes would make their votes one out of many, rather than the one and only.

The third approach would be to have one or more administrators keeping track of all ratings and comments going into the system, verifying the validity and quality of the comments prior to allowing other users to view them, as well as working on keeping the program database updated, giving expert advice on certain programs, such as well-known white listed applications, etc. However, once the number of users has reached a certain level, this would require a lot of manual work, which could become expensive for maintaining a free program, as well as seriously decrease the frequency of vote updates.

In addition to the problem with users that unintentionally provide the reputation system with incorrect information is the more complex threat by individuals, or groups of people, that decide to purposely abuse the systems. In the preventive anti-PIS reputation system, one such attack would be to intentionally try to enter a massive amount of incorrect data into the database. Either to slow the system down, or even crash it, or to target specific applications, trying to subject them to positive or negative discrimination. The main question when it comes to vote flooding is how to allow normal users to be able to vote smoothly and yet be able to address abusive users that attack the system.

An important aspect to take into consideration is that the server must ensure that each user only votes for a software program exactly once. A common solution to this kind of problem would be to let the user register a user account at the server before being able to activate the client software. For each user account, only one vote and comment can be registered for a specific software. Using some non-automatable process, such as image verification, and requiring a valid e-mail address during the registration of a new user account would help prevent the system for users trying to automatically create a number of new accounts to avoid the limit imposed on the number of votes each user can give to each software [10].

2.2 Protecting Users' Privacy

As the system is built for protecting peoples' privacy, we need to make sure the system itself does not intrude on it more than absolutely necessary. If the system would store sensitive information about its users, such as IP addresses, e-mail address, and linking these to all software the user has ever cast a vote on, the system owner would control this sensitive information. Any leakage of such information e.g., through an attack on the reputation system database, could have serious consequences for all users. An attacker getting access to this information would find a list of hosts and software running on each host, where some of them could be vulnerable to remote exploits. However, not storing any data about which users have cast votes on a particular software could lead to vote flooding and similar, as the system would have no way of ensuring that a user only votes once.

As we need to make sure no users can vote more than once on each particular software, we cannot get rid of the concept of users and user accounts. However, one approach would be to ensure that all kinds of sensitive information that can be of use for an attacker, such as IP address, e-mail address, name, address, city, or similar, are excluded from the user information stored in the database of the reputation system server. The only thing necessary to store is some kind of unique identifier for each user, such as a user name.

As mentioned in the previous section, we need to prevent users from signing up several times in an automatic way, and one way of doing this would be to use their e-mail address as an identification item. However, this requires us to store the e-mail address in the database which might not be something that people would like to store in a database that keeps track of which software they are running and their opinions on it. A solution to this would be to only keep a hash value of the e-mail address, as this can be used to discover that two e-mail addresses are equal, while it is impossible to recreate the e-mail address from the hash value. However, it would still be possible to guess the correct e-mail address if relying on a brute force approach. This problem could be further solved by concatenating the e-mail address with a secret string before calculating the hash, rendering brute force attack to be computationally impossible as long as the secret string is kept secret. Protection of users' anonymity could be established by utilizing distributed anonymity services, such as Tor, for all communication between the client and the server [36]. This would further increase user's privacy by their IP address from the reputation system owner.

3 System Design

As we have illustrated in the previous section, there are numerous aspects to take into consideration when designing a reputation system such as this. Information has to be gathered from the reputation system users in a way that address different ways of abuse, without interfering with normal usage and / or the protection of the users' privacy. When considering votes and comments, the system has to be able to handle possible abuse, as well as to properly balance the weight of different users' ratings and allow users to grade each others, thus improving the credibility of the more expert users and degrading users not taking voting and commenting in the system seriously.

The system will be comprised of three major parts, a client with a graphical user interface (GUI) running on each users' workstation, a server running on one or more machines handling all requests and commits from the clients, as well as a database storing all data. The system will also offer a web based interface, which gives the users more possibilities in searching the information stored in the database. This will be used as an extension to the GUI client, where users e.g. can read more information about some particular software program or vendor along with all the comments that have been submitted.

3.1 Client Design

The most important functionality of the client is the ability to allow its users to decide exactly what software is allowed to run on the computer, i.e. blocking all software which the user have not explicitly given his/her permission to. This filtering capability is implemented using a hooking device that captures the execution calls from the Windows API, in order to allow the user to choose whether or not he or she really wants to proceed with the execution of that particular software. Whenever software is trying to execute, the hooking device informs the client about the pending execution, which in turn asks the user for confirmation before actually running the software requesting to execute. The API hooking is used to capture the execution call that goes to the Windows kernel when the operating system tries to allocate memory for the program. We used Anton Bassov's Soviet Protector code when implementing the API hooking functionality, with slight modifications added [16]. It consists of a system driver that replaces the API call to `NtCreateSection()` with its own version, and a software component that communicates with the driver through a shared section of the memory[2].

The client uses different lists to keep track of which software have been marked as safe (the white list) and which have been marked as unsafe (the black list). These two lists are then used for automatically allowing or denying software to run, without asking for the user's permission every time, and thereby reducing the need for user interaction. When the driver discovers an execution and informs the client about it, the client traverses the white list and blacklist for an occurrence of the pending software based on a checksum calculated from the EXE-file content, using an algorithm such as for instance SHA-1. If the software is found in either of the two lists, the appropriate response is automatically sent to the driver without the need for user interaction, otherwise the client queries the server and fetches the information about the executing software to show the user and take action based on the user's decision.

[2] It might at first seem more reasonable to focus on API calls such as `NtCreateProcess()` [16].

The proof-of-concept tool also allows the user to submit ratings and comments, as described in the previous sections, as well as to view compiled information from other users and run statistics about the software about to execute. The user is only asked to rate software which he has executed more than a predefined number of times, currently 50 times. This ensures that the user has been using the software for some time and therefore has developed some sort of opinion about it. To minimize the user interruption there is also a threshold on the number of software the user is asked to rate each week, currently two ratings per week. So, when the user has executed a specific software 50 times she will be asked to rate it the next time it is started, unless two software already has been rated that week.

3.2 Server Design

In addition to the processing of software ratings the server also handles the database containing registered user information, ratings and comments for different software that users have previously voted on. The clients communicates with the server through a web-server that handles the requests sent by the client software, as well as displaying web pages for showing more detailed information about the software and comments in the database. XML is used as the communication protocol between the client and the server.

The only data stored in the database about the user is a username, hashed password and a hashed e-mail address, as well as timestamps of when the user signed up, and was last logged in. The e-mail address is only there to make it more difficult for a person to create several different accounts, as it is possible to sign up only once per e-mail address. Each e-mail address used to sign up must be valid, since it is used for the confirmation and activation of the newly created account.

From this data it is not possible for us, or anyone else getting in hold of the database, to identify a specific user, as long as the username (over the contents of which we have little control) does not reveal too much detailed information. And as our implementation does not store any IP addresses associated with the users, it is also impossible to determine which hosts are running which software, and from there try to launch an attack against a specific host. What can be traced however, is every user's submitted rating, comment and answers for each software he or she has ever rated, as well as each user's submitted remark (positive for a good, clear and useful comment or negative for a coloured, non-sense or meaningless comment) for every comment he or she has ever rated. But as mentioned previously, it is impossible to directly or indirectly associate this data with a particular host, but only to a username, hashed password, hashed e-mail address and two timestamps, which does not put the user at any actual risk from using this software.

Software ratings are calculated at fixed points in time (currently once in every 24-hour period). During this work users' trust factors are taken into consideration when calculating the final score for a particular software. In addition to these software ratings the proof-of-concept tool also calculates specific software vendor ratings. This is done by simply calculating the average score of all software belonging to the particular vendor.

As a protection mechanism, the reputation system has implemented a growth limitation on users' trust factors, by setting the maximum growth per week to 5 units.

Hence, you can reach a maximum trust factor of 5 the first week you are a member, 10 the second week, and so on. Thereby preventing any user from gaining a high trust factor and a high influence without proving themselves worthy of it over a relatively long period of time. The second limitation of the trust factor is a minimum level of 1 (which is also the rating for new users), and a maximum of 100.

3.3 Database Design

Each software represented in the database will hold a set of information that is linked directly to the executable file. The most important information is the unique software ID number which is generated by utilizing a hash algorithm over the file content. Since this ID is a calculated out of the file data (its program instructions) it is also directly connected to the software behaviour. This means that it is impossible to change the software behaviour without also changing the software ID. In other words, it is impossible to alter the programs behaviour and still keep the ratings associated with the software in the database, which is an important property for a software reputation system. Since the software ID is generated through by a hash algorithm (e.g. SHA-1) the risk of two different files having identical fingerprints is virtually non-existent. In addition to user ratings and comments the following information is stored for each software in the database:

- ID of software executable e.g., a generated SHA-1 digest.
- File name of the software executable.
- File size of the software executable.
- Company name of the software company that produced the software executable.
- Software version number.

Information about both the company name and file version is dependant on the software developer to put these values into the program file, which unfortunately is not always true. The rest of the data is meta-data that always can be retrieved once the complete file is in ones possession.

Since hash functions are used, the software ID will be different even between files with small modifications, in effect, two different versions of the same program will end up having different fingerprints. This also means they will be considered as separate software executables by the reputation system server, and as such their votes and ratings will be separated from each other. Although a drawback with this approach is that there will be many different database entries for slightly different versions of the same program, this may in fact be beneficial to the user. For example, one version of an application may be well known to cause degraded performance, display banners, and so on, while in the next version, the developers have fixed the performance issues and decided to use other means to finance their work, and thus the contents of the reputation system will correctly present this to the user.

However, questionable software vendors that want to try to circumvent the reputation system could try to make each instance of their software applications differ slightly between each other so that each one has its own distinct hash value. The countermeasure against such behaviour would be to instead map all ratings to the software vendor instead of mapping it to a specific software version from that vendor.

To fight that countermeasure some vendors might try to remove their company name from the binary files. If this should happen it could be used as a signal for PIS since legitimate software vendors label their products with their company information [6].

Furthermore, it would be possible for the system to provide users with valuable information about the vendor of a specific software by calculating the mean value over all software ratings the company in question has received. Giving the user an indication of how well the software developed by this company has previously fared in the reputation system. That way, the user may choose to base his decision on ratings and comments given not only on the current software executable, but also on the derived total rating of the software developing company.

4 Discussion

In this section we will discuss what impact the introduction of a software reputation system would have on privacy-invasive software. We will also bring up some issues with the proof-of-concept implementation together with improvement suggestions. In the end we make a comparison between existing countermeasures against PIS and the software reputation system.

4.1 System Impact

Offering users mechanisms that enhance informed decisions regarding software installation increase the liability of the user. In a way, these mechanisms transfer some of the responsibility concerned with the protection against PIS to the users themselves. We believe this is a necessary consequence for new protection mechanisms that respect users' own personal needs and preferences. As users are being confronted with descriptions about behaviours and consequences for PIS, they are also assumed to assimilate and use this information in a mature and reasonable way. Based on the reputation system, it would be up to the users themselves to decide on whether or not to allow certain software to enter their system.

Table 2. Difference between legitimate software and malware with respect to user's informed consent and negative user consequences

	Tolerable Negative Consequences	**Moderate Negative Consequences**	**Severe Negative Consequences**
High Consent	1) Legitimate software	2) Adverse software	3) Double agents
Low Consent	7) Covert software	8) Trojans	9) Parasites

Computer users today face similar difficulties when evaluating software as consumers did a hundred years ago when evaluating food products. In the nineteenth century food industry, distribution of snake-oil product flourished [35]. These products claimed to do one thing, for example to grow hair, while they instead made unwitting consumer addicted to habit-forming substances like cocaine and alcohol. In 1906 the Pure Food and Drug Act was passed by the United States Congress, allowing any manufacturer not complying with the rules to be punished according to the law [20]. As a consequence the manufacturers followed these rules, allowing consumers to trust the information on the food container to be correct. Further allowing them to make informed decisions on whether they should consume a product or not, based on individual preferences such as nutritiousness, degree of fat or sugar, price, or allergies. As long as the food does not include poisonous substances or use deceptive descriptions it is up to the consumer to make the final decision. Although the distribution of physical snake-oil products was mitigated in 1906, its digital counterpart continues to thrive under the buoyant concept of spyware. An important distinction between food products and software is that the former one relies on physical factories and companies with employed personnel, which software does not. It is possible for anyone with the programming skills to produce software which then is spread globally over the Internet. Since users do not always have the option to relate the software to a physical manufacturer we believe it is important for them to instead be able to use other users' previous knowledge about the product in question, offered to them by a software reputation system.

It should be noted that a reputation system against PIS tightly affect the PIS classification in Table 1. The introduction of this type of user-oriented countermeasure would transform the classification of PIS as shown in Table 2. As computer users are given a tool to make informed decisions regarding the behaviour and implications of software, it is possible to apply a sharp boundary based on user consent between all software in the PIS classification. Using the added knowledge provided by the reputation system would render in that all PIS that previously have suffered from a medium user consent level, now instead would be transformed into either a high consent level (i.e. legitimate software) or a low consent level (i.e. malware). In other words, all software with medium user consent, i.e. spyware, is transformed into either legitimate software or malware in the classification. Since anti-malware tools handle all malicious and deceitful software, the information about the rest of the software could be trusted to be correct, i.e. any software using deceitful methods is regarded as malware and are treated as such. This allows users to rely on the information when reaching trust decisions regarding software installation on their system. Another aspect of this type of countermeasure is that no single organization, company or individual is responsible for the software ratings, since these are calculated based on all votes submitted by the users. Making it hard for dissatisfied spyware vendors to sue the reputation system owners for defamation.

4.2 Improvement Suggestions

One issue that we soon discovered during tests of the proof-of-concept tool was the question of system stability. As we give the users the ability to deny the execution of important system components, we also handed them the ability to crash the entire system in a single mouse click. This further enhances the need for a white list system to ensure proper operating system functionality in order to avoid inadvertently bringing the operating system down when running the software client. However, given that the user has the free choice to block any program, there is no way to guarantee that the operating system will not be crashed, especially at an initial phase where the user is learning how to use the software client.

The proposed solution to this problem would be an enhanced white listing system that could examine the file about to execute, to determine if it has been digitally signed by a trusted vendor e.g., Microsoft or Adobe. In case the certificate is present and valid, the file is automatically allowed to proceed with the execution. It would also be possible to implement a signature handling interface in the reputation system client that allows the user to white list and blacklist different companies through their digital signatures, which – in turn – could considerably lower the need for user interaction. In that regard the proposed functionality would be somewhat similar to the capabilities of Microsoft's Restricted Software Policies [23].

The introduction of an enhanced white listing system with signature verification capabilities would provide an important building block for a software policy manager. By using the information available in the reputation system it would be possible for corporations or individual users to set up policies for what software is allowed to execute on their computers. Such policies could for instance take into account whether the software has been signed by a trusted vendor, the software and vendor rating, or any specific behaviour reported for the software e.g., if it show pop-up advertisements or include an incomplete removal routine. This would allow system owners to define policies for what software is allowed to install and run on their computers e.g., by specifying that any software from trusted vendors should be allowed, while other software only is allowed if it has a rating over 7.5/10 and does not show any advertisements. A solution like this implies that the reputation system also includes a preference module that holds the users' software preferences that should be enforced.

Another improvement suggestion involves allowing for instance organisations or groups of technically skilled individuals to publish their software ratings and other feedback within the reputation system. This information is then available for any other users of the reputation system. Allowing computer users to subscribe to information from organisations or groups that they find trustworthy, i.e. not having to worry about unskilled users that might negatively influence the information. The subscribed information could of course also be used in parallel with the other software feedback which is based on all reputation system members' votes.

4.3 Comparison with Existing Countermeasures

One major difference between traditional anti-spyware software and the reputation system based solution we propose is that in the latter we are able to gather more complete and useful information regarding the behaviour of software. Instead of a black and white world where an executable is branded as either a virus or not, we are able to touch the previously mentioned grey zone in between. We gather and present information about software that is important and useful to the users, and hard to find.

For instance, although an application may not be classified as a virus or spyware, users may think twice about running it if they are informed that it displays pop-up ads, registers itself as a start-up program and does not provide a functioning uninstall option. This kind of discouraging information will not be provided by the vendor of the application and can only be received from users who have experienced it first-hand and are willing to share their experiences to help others.

Currently available countermeasures against PIS, such as anti-spyware and anti-virus applications, have the benefit of specialized, up to date and reliable information databases that are updated on a regular basis. The drawback is a vendor database that must be updated locally on the client, as well as traversed whenever a file is analysed. Furthermore, the organization behind the countermeasure must investigate every software before being able to offer a protection against it. The relevance and reliability of the information provided by the anti-spyware and anti-virus software may be more reliable than that of users of a reputation system. However, the reputation system is able to cover more details that may be useful to the user, such as if the software displays ads, alter system settings, and so on, and with a sufficiently large user base, the sheer amount of data gathered helps compensate for the afore mentioned reliability issue. Also, by using a more flexible classification, where the user is provided the information about the software and is allowed to make an informed decision about allowing it to run or not, one is able to avoid the high contrast environment of anti-virus software and similar, where an executable is either strictly malicious or it is totally safe.

Different protection systems (e.g., anti-virus or anti-spyware tools) are built on different approaches, and the technology as well as pricing varies. In truth, it would be foolish to believe that either one approach would be a perfect solution to the problem at hand, and the view of the problem itself may differ. However, when looking at the development of the computer world, the Internet, and the on-going arms race in virus and spyware development, it is obvious that more than just one kind of protection is needed, and that there is no silver bullet. At the same time, we firmly believe that a specialized reputation system such as the one we propose would be a useful way to be able to penetrate the gray zone of half-legitimate software and to better inform users of what to expect from the software they are about to execute. It can be seen as trying to share and transfer knowledge between users, improving their level of expertise, instead of creating an expert system that handles all the decisions for the users, being ultimately responsible for the failure when the protection fails.

5 Conclusions and Future Work

This paper explores how to construct a specialized reputation system to be used for blocking privacy-invasive software. The fundamental idea is that computer users could be strong together if they collaborate to mitigate the effects from privacy-invasive software. The co-operation is based on that each users rate the software that they use most frequently. These aggregated ratings from the users are then transformed into software reputations that are available for all participants in the system upon installation of new software. Various methods to address incorrect information in the system, be it intentional or unintentional, are proposed without deteriorating users' privacy.

To further explore the possibilities, we designed and implemented a client and server-based proof-of-concept tool, which currently include well over 2000 rated software programs. Each time a user is about to execute a program, the client pauses the execution, downloads information and rating about the particular software from the server, and asks the user whether he or she would like to allow or deny the software to run. We further propose how this system could be enhanced by adding functionality that allows users to produce software policies that are automatically enforced by the client program. Such policies could take into account whether the software in question has received a rating above a certain value, if it is digitally signed by a trusted vendor, or if it is free from a set of predefined unwanted behaviours.

As future work we will investigate how and to what extent this proof-of-concept tool affects computer users' decisions when installing software. In addition to this we will also examine the possibility of using runtime software analysis to automatically collect information about whether software has some unwanted behaviour, for instance if it shows advertisements or includes an incomplete uninstallation function [24]. The results from such investigations could then be inserted into the reputation system as hard evidence on the behaviour for that specific software. Furthermore, it would be interesting to investigate the use of alternative, and more reliable, security tokens than e-mail addresses when creating new accounts. Maybe also relying on the IP address and computational penalties through variable hash guessing [3]. Finally, it would be interesting to investigate how pseudonyms could be used as a way to protect user privacy and anonymity, e.g. through the use of idemix [17].

References

[1] Ames, W.: Understanding spyware: risk and response. IEEE IT Professional 6(5) (2004)

[2] Arnett, K.P.: Busting the Ghost in the Machine. Communications of the ACM 48(8) (2005)

[3] Aura, T.: DOS-Resistant Authentication with Client Puzzles. LNCS, vol. 2133. Springer, Heidelberg (2000)

[4] Boldt, M.: Privacy-Invasive Software - Exploring Effects and Countermeasures, Licentiate Thesis Series No. 2007:01, School of Engineering, Blekinge Institute of Technology, Sweden (2007)

[5] Boldt, M., Carlsson, B.: Privacy-Invasive Software and Preventive Mechanisms. In: The proceedings of the IEEE International Conference on Systems and Networks Communications (ICSNC06), Papeete Tahiti, IEEE Computer Society Press, Los Alamitos (2006)

[6] Boldt, M., Carlsson, B., Martinsson, R.: Software Vulnerability Assessment - Version Extraction and Verification. In: The proceedings of the Second International Conference on Software Engineering Advances (ICSEA'07), Cap Esterel France (2007)

[7] Bruce, J.: Defining Rules for Acceptable Adware. In: The Proceedings of the 15th Virus Bulletin Conference. Dublin Ireland (2005)

[8] Christodorescu, M., Jha, S.: Testing Malware Detectors. In: The proceedings of the ACM International Symposium on Software Testing and Analysis (2004)
[9] Dellarocas, C.: Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behaviour. In: The proceedings of the 2nd ACM Conference on Electronic Commerce (2000)
[10] Douceur, J.: The Sybil Attack. In: The proceedings for the 1st International Workshop on Peer-to-Peer Systems (2002)
[11] Fischer-Hübner, S.: IT-Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms. Springer, Heidelberg (2001)
[12] Flixster (September 13, 2006), http://www.flixster.com
[13] Good, N., et al.: Stopping Spyware at the Gate: A User Study of Privacy, Notice and Spyware. In: The proceedings of the Symposium on Usable Privacy and Security, Pittsburgh, USA (2005)
[14] Good, N., et al.: User Choices and Regret: Understanding Users' Decision Process about Consentually Acquired Spyware. I/S: A Journal of Law and Policy for the Information Society 2(2) (2006)
[15] Greenfield, A.: Everyware - The Dawning Age of Ubiquitous Computing. New Riders, Berkeley CA (2006)
[16] Hooking the native API and controlling process creation on a system-wide basis (November 23, 2006), http://www.codeproject.com/system/soviet_protector.asp
[17] Idemix: pseudonymity for e-transactions (June 28, 2006), http://www.zurich.ibm.com/security/idemix/
[18] Internet Movie Database (February 23, 2007), http://www.imdb.com
[19] Kirda, E., Kruegel, C., Banks, G., Vigna, G., Kemmerer, R.A.: Behavior-Based Spyware Detection. In: The proceedings of the 15th USENIX Security Symposium (2006)
[20] Landmark Document in American History, Pure Food and Drug Act of 1906 (October 16, 2006), http://coursesa.matrix.msu.edu/~hst203/documents/pure.html
[21] Larsson, T., Lindén, N.: Blocking Privacy-Invasive Software Using a Specialized Reputation System, Masters Thesis No. 2006:14, School of Engineering, Blekinge Institute of Technology, Sweden (2006)
[22] LavaSoft Ad-Aware (September 19, 2006), http://www.lavasoftusa.com/software/adaware
[23] Microsoft Technet, Using Software Restriction Policies to Protect Against Unauthorized Software (May 13, 2007)
[24] Moshchuk, T., Bragin, S.D., Gribble, H.M.: A Crawler-based Study of Spyware on the Web. In: The proceedings of the Network and Distributed System Security Symposium Conference Proceedings, Virginia USA (2006)
[25] Norton Internet Security (September 19, 2006), http://www.symantec.se/region/se/product/nis index.html
[26] Pricerunner (September 19, 2006) http://www.pricerunner.com
[27] Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation Systems. Communications of the ACM 42(12) (2000)
[28] Rosenberg, R.S.: The Social Impact of Computers, 3rd edn., San Diego CA. Elsevier Academic Press, Amsterdam (2004)
[29] See you later - anti-Gators, CNET News.com (September 19, 2006), http://news.com.com/2100-1032 3-5095051.html
[30] Schultz, K.: Sticking It to Spyware. InfoWorld 27(38) (2005)

[31] Sipior, J.C.: A United States Perspective on the Ethical and Legal Issues of Spyware. In: Proceedings of 7th International Conference on Electronic Commerce, Xi'an China (2005)

[32] Spyaudit (September 12, 2006), http://www.earthlink.net/spyaudit/press/

[33] Spybot -Search & Destroy (September 19, 2006), http://www.safer networking.org

[34] "Spyware": Research, Testing, Legislation, and Suits (March 01 2006) http://www.benedelman.org/spyware/

[35] Technology Review, The Pure Software Act of 2006 (October 16, 2006), http://www.simson.net/clips/2004/2004.TR.04.PureSoftware.pdf

[36] Tor: anonymity online (February 24, 2007), http://tor.eff.org

[37] Webroot Software, —.: Internet Spyware and statistics about infection rate (September 12, 2006), http://www.webroot.com/resources/stateofspyware/excerpt.html

[38] Zacharia, G., Moukas, A., Maes, P.: Collaborative Reputation Mechanisms in Electronic Marketplaces. In: the proceedings of the 32nd Hawaii International Conference on System Sciences (1999)

Towards Improved Privacy Policy Coverage in Healthcare Using Policy Refinement

Rafae Bhatti and Tyrone Grandison

IBM Almaden Research Center,
650 Harry Road, San Jose, California 95120, USA
{rbhatti,tyroneg}@us.ibm.com

Abstract. It is now mandatory for healthcare organizations to specify and publish their privacy policies. This has made privacy management initiatives in the healthcare sector increasingly important. However, several recent reports in the public media and the research community about healthcare privacy [1,2] indicate that the use of privacy policies is not necessarily a strong indication of adequate privacy protection for the patient. These observations highlight the fact that the current state of privacy management in healthcare organizations needs improvement. In this paper, we present PRIMA, a PRIvacy Management Architecture, as a first step in addressing this concern. The fundamental idea behind PRIMA is to exploit *policy refinement* techniques to gradually and seamlessly embed privacy controls into the clinical workflow based on the actual practices of the organization in order to improve the *coverage* of the privacy policy. PRIMA effectively enables the transition from the current state of *perceived to be privacy-preserving* systems to *actually privacy-preserving* systems.

Keywords: Privacy Management, Healthcare, HIPAA, Compliance, Refinement.

1 Introduction

Privacy management is one of the main inhibitors of the deployment, adoption and use of electronic records systems in the healthcare industry. There are several privacy laws and regulations that have emerged around the world in the past few years [3], such as the Personal Data Protection Law [4] in Japan, the Health Insurance Portability and Accountability Act (HIPAA) in the United States [5] and the Personal Information Protection and Electronic Documents Act [6] in Canada. For American healthcare, HIPAA is normally assumed to provide the baseline for privacy compliance for healthcare entities.

While HIPAA and other healthcare-related privacy laws and regulations make it mandatory for organizations to specify and publish privacy policies regarding the use and disclosure of personal health information, recent media and academic reports about healthcare privacy [1,2] indicate that there is not necessarily a strong correlation between the use of privacy policies and adequate patient privacy protection. In [2], the authors examined the actual access patterns for a Norwegian healthcare organization. Their study indicates that in spite of possessing strict policy and regulation, the security mechanisms of the IT system were under-utilized and often bypassed in order to deliver care. This phenomenon creates an over-reliance on exception-based access for any situation that does not seem to be explicitly covered by the policy, and even for some that are. In the healthcare environment, disallowing access during service delivery is not an option because it may lead to grim consequences for the patient.

These observations, which have also been echoed in the United States [1,7], intimate the existence of a general state of affairs in healthcare organizations, where circumventing data security and privacy controls is the rule and not the exception. This trend is alarming because it negates the existence and efficacy of policy. In this state, the policy does not precisely represent or embody the actual level of data protection afforded to the patient, i.e. the policy is no longer a genuine reflection of the organization's privacy practices. Additionally, it undermines the notion of empowering the patient, as his consent may no longer be valid because the policy is no longer valid. While these observations may appear to reflect negatively on the healthcare organizations, these scenarios are in fact a direct result of applying prior technology without considering the nuances of the clinical workflow. In light of the recent push to electronic health records [8], this conundrum will multiply in effect.

It is our belief that it is possible to leverage artifacts from the actual clinical workflow to inform and construct appropriate privacy protection mechanisms for patients. We purport that *policy refinement*, which we will define as the process of improving the rules that define the level of protection, can be employed to gradually and seamlessly embed meaningful privacy controls into the clinical workflow based on the actual practices of the organization. This concept is the base construct for the PRIMA system, which also leverages data mining [9] and Hippocratic Database technology [10]. In particular, the architecture builds upon the *Active Enforcement* [11] and *Compliance Auditing* [12] components of the Hippocratic Database technology, and leverages standard data analysis techniques. PRIMA's *policy refinement* helps mitigate the above stated conundrum by (i) improving the design of the policies, which should elevate the level of privacy protection afforded to the patient, and (ii) better aligning the system policies with the actual privacy practices of the organization to improve the *coverage* of the privacy policy. To the best of our knowledge, no prior work on policy refinement for healthcare systems has been undertaken.

The rest of the paper is organized as follows. In section 2, we provide some background from a regulatory standpoint regarding use and disclosure of personal health information. Then we analyze the rationale for stated privacy policies not being actual representations of patient privacy protection. In section 4, we describe the PRIMA architecture and technical details. In section 5, we illustrate the use of PRIMA in a healthcare scenario. We conclude in section 6.

2 Background

Privacy legislation around the world are based on the notions captured in the OECD Data Protection Principles [3]. For the purposes of exemplification, and without loss of generality, we ground our discussion on privacy protection in the healthcare sector by examining the **Limited Use and Disclosure** provision of the HIPAA Privacy Rule. The motivation and arguments for this provision can be extrapolated to the other similar legislation, regulations and laws around the world.

With respect to the HIPAA Privacy Rule, *covered entities* refer to health plans, healthcare providers and healthcare clearinghouses, and *Protected Health Information (PHI)* refers to all individually identifiable health information held or transmitted by a covered entity or its business associate, electronically, on paper, or orally. The limited use and disclosure provision requires that covered entities must use or disclose the *minimum necessary* PHI for a specific purpose and ensure the development and implementation of policies and procedures governing access and use.

In accordance with the purpose specification provision in privacy regulations, a privacy policy statement normally contains specific purposes for which data can be used or disclosed. However, the defined purposes tend to be very broad in scope [2]. For example, many real-world policies mention collecting information for the purpose of "administering healthcare". This granularity is coarse enough to subsume many information uses and disclosures. We recognize that this practice may not be performed with mal-intent, but may be a function of reducing the complexity of policy specification, which reduces the size of the rule base.

It was also observed [2] that organizations had difficulty defining specific employee categories (i.e. useful roles), which define the authorizations for viewing specific patient data categories [13]. Typically, the collected information is available to all "members of medical staff", which effectively results in an umbrella authorization. Again, we recognize that the transition from the *generalist school of medicine* to the *specialist school of medicine* over the last few decades has meant that the number of healthcare professionals involved in the delivery of care, to a single patient, has increased significantly and that categorizations may be hard because roles are so fluid and cannot be assumed to be mutually exclusive. Additionally, the primary purveyors of healthcare tends to be the nursing staff and it is understandable that authorization difficulties may exist. However, there are still clearly defined lines, at least legally, on who should be able to view and use particular aspects of patient data. Thus, role delineation and categorization is necessary and critical. For a few years now, the broader community has realized and advocated the need for fine-grained access control. This view is shared by both academic researchers [14,15,16] and medical professionals [17,13].

From the previous discussions, it is clear that, despite the underlying reasons, the limited use and disclosure provision in the HIPAA Privacy Rule has not been interpreted and implemented very well in existing healthcare informatics systems. Our view is that healthcare is an industry that will always require customized mechanisms to balance privacy and operational considerations.

In order to not be disruptive and to automate this process of customization, PRIMA attempts to gradually embed policy controls in the system by analyzing the information already existing in the system and informing the new state that the system needs to evolve to. Thus, the overall goal is to bridge the disparity between intended and achieved levels of privacy protection.

3 Formal Model

For an arbitrary healthcare organization, *HO*, the policy that they define for their IT systems embodies the regulations, legislation, laws and organizational

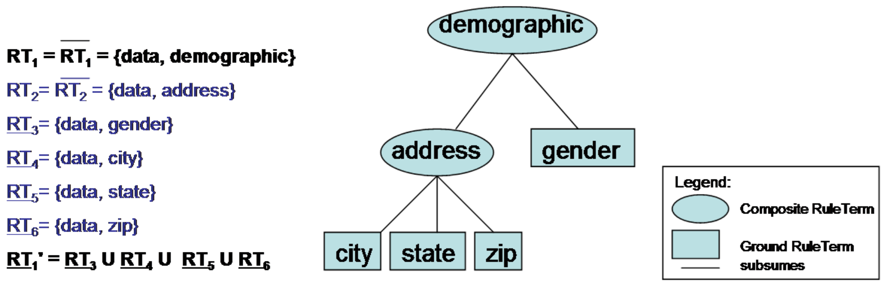

$RT_1 = \overline{RT_1}$ = {data, demographic}
$RT_2 = \overline{RT_2}$ = {data, address}
$\underline{RT_3}$= {data, gender}
$\underline{RT_4}$= {data, city}
$\underline{RT_5}$= {data, state}
$\underline{RT_6}$= {data, zip}
$\underline{RT_1}' = \underline{RT_3} \cup \underline{RT_4} \cup \underline{RT_5} \cup \underline{RT_6}$
demographic
address
gender
city
state
zip
Legend:
Composite RuleTerm
Ground RuleTerm
subsumes

Fig. 1. A Sample Privacy Policy Vocabulary

mandates that they must follow. This represents what they would ideally like to happen, i.e. their ideal workflow W_{Ideal}.

The studies presented earlier state that after a period of operation, the audit trails of system accesses, which represents *HO's* real workflow W_{Real}, is primarily filled with exception-based access statements.

3.1 Core Constructs

Let's formalize the underlying notions and the goal of the PRIMA system, which is the reduction of the gap between real and ideal workflows. We assume that the HO has chosen a privacy specification notation and has a mapping from the terms used in this notation to the artifacts that the IT system will manipulate. Hereafter, we refer to these artifacts as the privacy policy vocabulary (or vocabulary, for short). The formal representation of the policies relies on the key concepts that make up a policy, namely *RuleTerm* and *Rule*.

Definition 1. *(RuleTerm): A RuleTerm (RT) is a tuple with two literal-valued elements, attr and value. It is written as* $RT = (attr, value)$. *The two elements of RT are accessed as RT.attr and RT.value.* □

A *RuleTerm* models the assignment of an attribute in a policy rule. For example, demographic data is represented as (data, demographic) and telemarketing purposes as (purpose, telemarketing). *RuleTerm* is the fundamental construct for our formalism in order to ensure that the model is applicable to any arbitrary specification notation.

Definition 2. *(RuleTerm Types): A RuleTerm, RT, is considered* ***ground*** *(written as* $\underline{RT}$*) iff its attribute value (RT.value) is an atomic-valued literal, with respect to the privacy policy vocabulary used. Otherwise, it is* ***composite*** *(written as* $\overline{RT}$*).* □

Let's define RT_1 = (data, demographic), RT_2 = (data, address), and RT_3 = (data, gender). The particular policy vocabulary used here is depicted in Figure 1[1]. In this example, RT_3 can be considered a ground *RuleTerm* since it

[1] Only the *RT.value* element of each RT is shown in the figures for conciseness.

contains the attribute value "gender", which cannot be further divided into multiple $RuleTerm$s according to the chosen vocabulary. On the other hand, RT_1 is unequivocally a composite $RuleTerm$ since demographic information could be further divided into information about address and gender. In fact, both RT_2 and RT_3 are subsumed by RT_1.

For each composite RuleTerm $\overline{RT}$, we assume the existence of a special set, written as $\underline{RT'}$, that contains all the ground rule terms $\underline{RT_1}, \ldots, \underline{RT_n}$ that can be derived from $\overline{RT}$ using the chosen privacy policy vocabulary. In the example shown in Figure 1, the set $\underline{RT'_1}$ for $\overline{RT_1}$ is shown to comprise of four ground RuleTerms.

Definition 3. *(Existence of Ground RuleTerm): Given a composite RuleTerm $\overline{RT}$ and a privacy policy vocabulary, it can always be transformed to a corresponding ground RuleTerm $\underline{RT}$. Formally,* $(\forall x : RT(x \in \overline{RT})) \rightarrow (\exists y : RT(y \in \underline{x'})$.[2]

An important notable notion is that of the equivalence of $RuleTerm$s. Given a privacy policy vocabulary or set of vocabularies, the equivalence notion allows for the comparison of $RuleTerm$s.

Definition 4. *(Equivalence of RuleTerms): Two RuleTerms, RT_i and RT_j, are considered equivalent, written $RT_i \approx RT_j$, iff $\exists x, y : RT(x \in \underline{RT_i}) \wedge (y \in \underline{RT_j})$ $\wedge$ $(x.attr = y.attr) \wedge$ $(x.value = y.value))$.* □

In the example in Definition 1, both RT_2 and RT_3 are equivalent to RT_1 because there exists ground $RuleTerms$ $\underline{RT_2}$ and $\underline{RT_3}$ belonging to the set $\underline{RT'_1}$.

Definition 5. *(Rule): A Rule, R_i, is a conjunction of RuleTerms. It is written as $R_i = \{RT_1 \wedge \ldots \wedge RT_n\}, n \geq 1$. The number of RuleTerms of a Rule, n, is referred to as the cardinality of the Rule, written as $\#R$.* □

A $Rule$ models a specific combination of attribute assignments, which represents individual statements in a policy. For example, "nurses are authorized to see insurance information for billing purposes"may be represented as $\{(data, insurance) \wedge (purpose, billing) \wedge (authorized, nurse)\}$.

A $Rule$, R_i, is said to be a **ground** rule (written as $\underline{R_i}$) if all $RuleTerm$s in R_i are ground. R_i is a **composite** rule (written as $\overline{R_i}$) if there exists at least one $RuleTerm$ that is a composite $RuleTerm$.

Corollary 1. *(Existence of Ground Rule): From Definition 3, it follows that for any Rule R_i, there always exists a corresponding Rule $\underline{R_i}$.*

Definition 6. *(Equivalence of Rules): Two Rules, R_1 and R_2, are said to be equivalent, written as $R_1 \approx R_2$, iff $(\#\underline{R_1} = \#\underline{R_2}) \wedge (\forall x : RT\ (x \in \underline{R_1}) \rightarrow (\exists\ y : RT\ (y \in \underline{R_2}) \wedge (x \approx y))$.* □

Essentially, rules are equivalent when they have the same number of terms and every term in one rule is equivalent to another in the other rule.

Definition 7. *(Policy): A policy, P_x, is a collection of rules that is symbolically tied to a data store x, where x can be either the policy store, PS, or the audit logs, AL. A policy is written as $P_x = R_x^1, \ldots, R_x^m, m \geq 1$. The number of Rules in the P_x, m, is referred to as the cardinality of P_x, which is written as $\#P_x$.* □

[2] Note that $\underline{x'}$ is a set.

For our purposes, we equate W_{Ideal} to P_{PS} and W_{Real} to P_{AL}. This is a simplification that holds true because the artifacts under investigation are the workflows relating to healthcare data disclosure and use.

Given that $RuleTerm$s and $Rule$s can be either ground or composite, a $Policy$ can be too. A $Policy$, P_x, is a **ground** policy (written $\underline{P_x}$) if all $Rules$ are ground $Rule$s. If there is at least one composite $Rule$ in P_x, then it is a composite policy (written $\overline{P_x}$).

For each composite policy $\overline{P_x}$, we assume the existence of a special set, written as $\underline{P_x{}'}$, that contains all the ground rules that can be derived from the composite rules in P_x using the chosen privacy policy vocabulary. The existence of this set follows from Definitions 3, 5, and 7.

Corollary 2. *(Existence of Ground Policy): From Corollary 1, it follows that for any Policy P_x, there always exists a corresponding Policy $\underline{P_x}$.*

3.2 Policy Coverage

The concept of policy coverage builds upon the idea of comparing the real state of the system P_{AL}, as represented by the audit logs, with the ideal state of the system P_{PS}, as represented by the policy store that contains the rules specified by some system administrator, privacy officer, etc. We recognize that the P_{PS} will normally be specified at a high level of abstraction (and later mapped to low-level control statements), and that P_{AL} will be low-level information gathered by the system in its normal operation. Thus, in order to perform a meaningful comparison, we must transform both to the lowest common denomination, i.e ground policies, and then do our evaluation.

Definition 8. *(Range): Given a policy P_x, its range, $Range_{P_x}$, is the set containing all the rules in $\underline{P_x{}'}$.* □

The cardinality of the $Range$ is the number of elements in the set $Range_{P_x}$, written $\#Range_{P_x}$. Given this definition, we can now define policy coverage.

Definition 9. *(Coverage): Given two policies P_x and P_y, and a privacy policy vocabulary V, the coverage of P_x in relation to P_y, written $Coverage^{P_x}_{P_y}$, is given by $\#(Range_{P_x} \cap Range_{P_y}) \div \#Range_{P_y}$.* □

Here, the intersection is computed using the equivalence of rules as defined in Definition 6. Informally, the policy coverage in a given system is defined as the amount of overlap between the real and ideal representations of the system state, namely P_{PS} and P_{AL}. The coverage of P_x with respect to P_y is computed as a ratio using the algorithm ComputeCoverage given below.

The overall goal of the PRIMA system is to move towards a state of complete coverage, which is defined below. It is acknowledged that complete coverage may not be attainable given the human component, but higher levels of coverage should be a realistic goal. The process of improving the policy coverage is visually shown in Figure 2.

Definition 10. *(Complete Coverage): Given two policies P_x and P_y, and a privacy policy vocabulary V, P_x completely covers P_y iff $Range_{P_x} \cap Range_{P_y} = Range_{P_y}$.* □

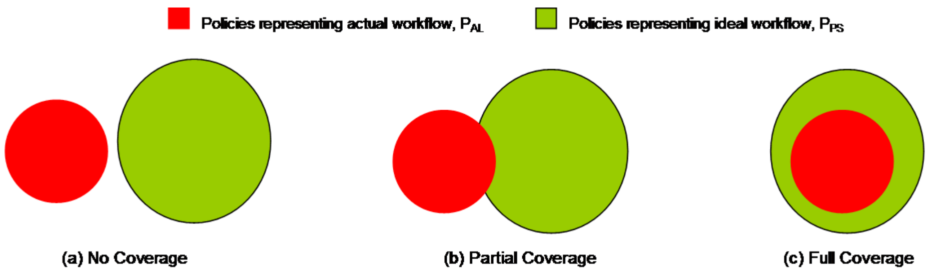

🟥 Policies representing actual workflow, P_{AL}
🟩 Policies representing ideal workflow, P_{PS}
(a) No Coverage
(b) Partial Coverage
(c) Full Coverage

Fig. 2. Simplified Visual Representation of Policy Coverage

Algorithm 1. $ComputeCoverage(P_x, P_y, V)$

```
Require: $\exists getCardinality(S)$ (returns the cardinality of a set S)
Require: $\exists getRange(P, V)$ (returns the range of the policy $P$ according to the policy
    vocabulary $V$)
1: $coverage \leftarrow 0$
2: $range_x[] \leftarrow getRange(P_x, V)$
3: $range_y[] \leftarrow getRange(P_y, V)$
4: $m_y \leftarrow getCardinality(range_y)$
5: $overlap[] \leftarrow range_x \cap range_y$
6: $m_o \leftarrow getCardinality(overlap)$
7: $coverage = m_o \div m_y$
8: return $coverage$
```

3.3 Illustrative Example

Let's look at a simple example that demonstrates coverage calculation. Consider the policy store shown in Figure 3(a). Let the policy tied to this policy store be denoted as P_{PS}. The top table shows the abstract-level composite policy, $\overline{P_{PS}}$, which comprises of three rules. The bottom table shows a portion of the ground policy, $\underline{P'_{PS}}$.

Now consider the audit logs shown in Figure 3(b). Let the policy tied to the audit logs be denoted as P_{AL}. By default, this policy is a ground policy, $\underline{P_{AL}}$, and it comprises of six rules. We observe that rules 1, 2, and 5 in $\underline{P_{AL}}$ are matched by rules 1a, 1b, and 3a, respectively, in $\underline{P'_{PS}}$, but rules 3, 4, and 6 in $\underline{P_{AL}}$ are not matched by any rules in $\underline{P'_{PS}}$. This indicates that exception-based accesses were utilized to access the data in a situation which not was allowed by the policy. These exception scenarios are pointed out in the figure.

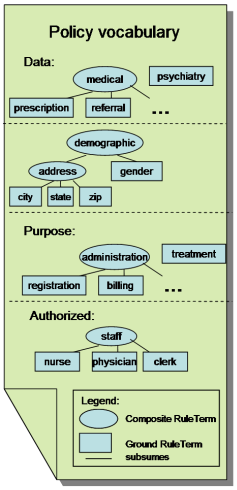
To elaborate, the reason for rule 3 not being matched is that a *nurse* needed to access *referral* data for *registration* purpose, but the policy allows the use of such data only for *treatment* purpose. The reason for rule 4 not being matched is that a *nurse* needed to access *psychiatry* data for *treatment* purpose, but the policy allows such data to be accessed only by a *physician*. Lastly, the reason for rule 6 not being matched is that a *clerk* needed to access *prescription* data for *billing* purpose, but the policy allows the use of only *demographic* data for this

$\overline{P}_{PS}$ (The abstract level composite policy tied to policy store)

Rule 1	Attr: data Value: medical	Attr: purpose Value: treatment	Attr: authorized Value: nurse
Rule 2	Attr: data Value: psychiatry	Attr: purpose Value: treatment	Attr: data Value: physician
Rule 3	Attr: data Value: demographic	Attribute: purpose Value: administration	Attribute: data Value: staff

P'_{PS} (The ground policy tied to policy store)

Rule 1a	Attr: data Value: prescription	Attr: purpose Value: treatment	Attr: authorized Value: nurse
Rule 1b	Attr: data Value: referral	Attr: purpose Value: treatment	Attr: authorized Value: nurse
Rule 2	Attr: data Value: psychiatry	Attr: purpose Value: treatment	Attr: authorized Value: physician
Rule 3a	Attr: data Value: address	Attr: purpose Value: billing	Attr: authorized Value: clerk
Rule 3b	Attr: data Value: state	Attr: purpose Value: billing	Attr: authorized Value: clerk
Rule 3c	Attr: data Value: zip	Attr: purpose Value: billing	Attr: authorized Value: clerk
Rule 3d	Attr: data Value: city	Attr: purpose Value: billing	Attr: authorized Value: clerk
Rule 3e	Attr: data Value: address	Attr: purpose Value: registration	Attr: authorized Value: nurse
...	...	...	...



(a)Policy tied to policy store

P_{AL} (The ground policy tied to audit logs)

Rule 1	Attr: data Value: prescription	Attr: purpose Value: treatment	Attr: authorized Valu: nurse
Rule 2	Attr: data Value: referral	Attr: purpose Value: treatment	Attr: authorized Value: nurse
Rule 3	Attr: data Value: referral	Attr: purpose Value: **registration**	Attr: authorized Value: nurse
Rule 4	Attr: data Value: psychiatry	Attr: purpose Value: treatment	Attr: authorized Value: **nurse**
Rule 5	Attr: data Value: address	Attr: purpose Value: billing	Attr: authorized Value: clerk
Rule 6	Attr: data Value: **prescription**	Attr: purpose Value: billing	Attr: authorized Value: clerk
...	...	...	...

Exception Scenarios:

"nurse" may need to access **"referral"** data for **"registration"** purposes too

"nurse" too may need to access **"psychiatry"** data for **"treatment"** purposes

"clerk" may need to access **"prescription"** data too for **"billing"** purposes

(b)Policy tied to audit logs

Fig. 3. Example scenario illustrating coverage computation

purpose. These scenarios indicate the customary practices during the clinical workflow which should be incorporated in the privacy policy of the system.

Invoking *Compute Coverage*(P_{PS},P_{AL},V) on this system, the policy coverage of P_{PS} with respect to P_{AL} in this system is found to be 50 %, i.e. $\#(Range_{P_{PS}} \cap Range_{P_{AL}}) \div \#Range_{P_{AL}}$ is 3/6.

4 PRIMA: The System

The discussion on policy coverage is useful in formally understanding the goal of PRIMA. However, a significant consideration, from the clinical standpoint, is the design of the PRIMA system in a way that aligns with, and not impedes, the clinical workflow. PRIMA attempts to improve policy coverage by gradually embedding new policy statements, which were discovered through the process of policy refinement, into the clinical system.

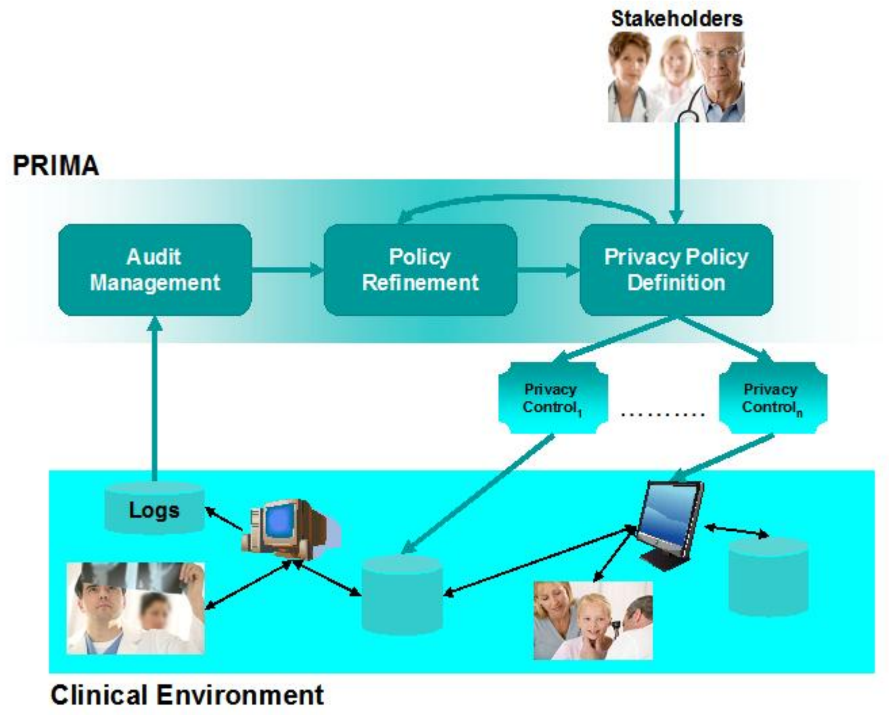


Fig. 4. The PRIvacy Management Architecture (PRIMA)

Figure 4 provides a high-level view of PRIMA. *Stakeholders* define the privacy policies for the *HO*, which is embedded in *privacy controls* that are integrated into the clinical environment. One of these privacy controls is an auditing function that automatically generates entries for the system's audit logs. These logs are either periodically replicated or PRIMA-enabled, by the construction of a consistent consolidated view of them. In the simplest case, there is just one log. We will discuss desired features of audit controls in section 4.2. At regular intervals or at the request of the stakeholders, the *Policy Refinement* component extracts input from the *Audit Management* component and the *Privacy Policy Definition* component and outputs a list of definitions, if any exist, that should be included in the policy definitions. Let's discuss each of these components in more depth.

4.1 Privacy Policy Definition

In this context, we assume that input is gathered by all the *stakeholders*, i.e. patients, medical practitioners, payers etc., and a representative uses this information to specify the *HO*'s policy. At an abstract level, PRIMA may leverage any arbitrary privacy policy definition tool that has the facility to create privacy controls that can be embedded into the clinical workflow. As a proof of concept, the initial instantiation utilizes the HDB Active Enforcement [11] and HDB

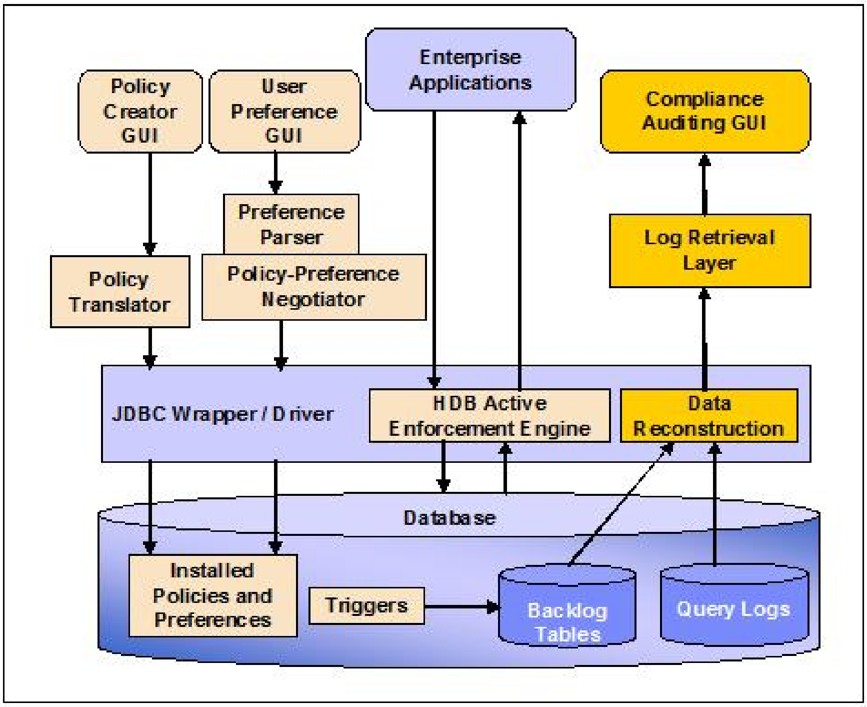


Fig. 5. Combined Architecture of HDB Active Enforcement (AE) and Compliance Auditing (CA)

Compliance Auditing [12] components (Figure 5), which produces augmented database interfaces that both enforce fine-grained policy and patient consent and create minimal impact, storage and performance efficient logs. Our user would use the HDB Control Center to enter fine-grained rules, patient consent information and specify what needs to be auditable.

The HDB components (Figure 5) operate at the middleware layer between the clinical database and the end user query interface. When the AE component receives user queries, it rewrites the queries so that only data consistent with policy and patient preferences is returned. The rewritten request gets sent to the database for execution and is also stored along with the query issuer, purpose, time and date in the audit log.

4.2 Audit Management

Retroactive controls, such as audit trails, and the threat of inevitable violation detection and prosecution are prevalent in healthcare information systems. Unfortunately, there are a series of concerns that may stem this approach. The first concern is the impact on the existing infrastructure, i.e. the degradation in system performance and the increased storage demand. The second is the nature of the technology's use, i.e. the logs tend to be used only when someone raises a red flag about an improper data disclosure, not as a part of a continuous, proactive process. Finally, not all the necessary contextual information may have been logged with the request. The first and third concerns translate into requirements for auditing systems within the clinical environment.

Use of HDB Compliance Auditing in the clinical workflow allows us to meet these two requirements. The schema for an audit entry is $\{$*(time,t_j), (op,X_j), (user,u_j), (data,d_j), (purpose,p_j), (authorized,a_j), (status,s_j)*$\}$, where t_j is the entry's timestamp, X_j is either 0 (disallow) or 1 (allow), u_j is the entity that requested access, d_j is the data to be accessed, p_j is the purpose for which the data is accessed, a_j is the authorization category (e.g. role) of the entity that requested access, and s_j is either 0 (exception-based access) or 1 (regular access). The status s_j of access would in practice be recorded at the time the user either chooses or manually enters the purpose of access, where former corresponds to a regular access and latter to an exception-based access. We realize that this model could be augmented with the inclusion of conditions. However, the techniques that will be used on the core elements presented are also applicable to augmentations of the model.

The PRIMA *Audit Management* component acts as a consolidation for the audit systems in the clinical environment. In the first instantiation, we use DB2 Information Integrator as the federation technology in the PRIMA *Audit Management* component to create a virtual view of all the audit trails. Alternative methods may be used that can consolidate all audit data in one place for subsequent analysis.

Irrespective of the mechanism used to populate the P_{AL} used by PRIMA, we must be cognizant that the audit logs may contain different kinds of information. There may be data on attempts to break into the system, i.e. possible violations or data breaches, or information that represents undocumented, informal clinical practice. We need to differentiate between violations and informal practice entries in the refinement process.

4.3 Policy Refinement

Refinement is based on the premise that a feedback loop is required between real and ideal policy; in order to create policy that (i) more accurately represents the covered entity's intent and behavior, and (ii) more adequately represents the level of privacy protection afforded to the patient.

The pseudocode for the refinement process is given in Algorithm 2. The function is provided (i) the policy store, P_{PS}, (ii) the policies in the logs, P_{AL}, and (iii) the privacy vocabulary, V, being used by this particular covered entity. P_{AL} is filtered to remove prohibitions, analysis is performed on the resulting set to create a set of patterns (if any exist), which are then pruned based on the coverage of P_{PS} with respect to P_{AL}.

Even though refinement is an ongoing process, we assume that there is a *training period*, where a reasonable amount of information is collected in the audit log. This training period is totally dependent on the particular healthcare entity deploying the system.

Filter. Algorithm 3 outlines the filter process. Given the schema in subsection 4.2 and the policy under examination, this process removes all rules that are

Algorithm 2. $Refinement(P_{PS}, P_{AL}, V)$

```
Require: $\exists Filter(P)$ (returns the non-prohibitions in policy P)
Require: $\exists extractPatterns(P)$ (returns the rules that may be undocumented patterns)
Require: $\exists Prune(Patterns, P_{PS}, V)$ (returns the patterns to be incorporated into the system's current policy)
1: $Practice[] \leftarrow$ Filter($P_{AL}$) (see Algorithm 3)
2: $Patterns[] \leftarrow$ extractPatterns($Practice, V$) (see Algorithm 4)
3: $usefulPatterns[] \leftarrow$ Prune($Patterns, P_{PS}, V$) (see Algorithm 6)
4: return $usefulPatterns$
```

Algorithm 3. $Filter(P)$

```
Require: $\exists getCardinality(P)$ (returns the cardinality of P)
Require: $\exists getRule(P, i)$ (returns the $i$th rule of policy P)
Require: $\exists getStatus(R)$ (returns the value for the status attribute in rule R)
Require: $\exists appends(R, R_{set})$ (appends Rule $R$ to the set of rules $R_{set}$)
1: $Practice \leftarrow []$
2: $n \leftarrow getCardinality(P)$
3: for $i = 1$ to $n$ do
4:    $R_i \leftarrow getRule(P, i)$
5:    if $getStatus(R_i) == 0$ then
6:       $append(R_i, Practice)$
7:    end if
8: end for
9: return $Practice$
```

Algorithm 4. $extractPatterns(P, V)$

```
Require: $\exists dataAnalysis(P, A, f, c)$ (Given a policy $P$, an Audit Schema (or a subset thereof) $A$, a frequency $f$ and a condition $c$, perform data analysis)
1: $A \leftarrow$ get attributes from Audit Schema (may also be sent to any subset of Audit Schema)
2: $f \leftarrow$ system-defined threshold frequency (by default set to 5)
3: $c \leftarrow$ system defined condition (by default set to $COUNT(DISTINCT(User) > 1)$
4: $Patterns \leftarrow []$
5: $Patterns[] \leftarrow$ dataAnalysis($P, A, f, c$)(see Algorithm 5)
6: return $Patterns$
```

Algorithm 5. $dataAnalysis(P, A, f, c)$

```
Require: $\exists executeQuery(SQL)$ (executes $SQL$ statement and returns results)
1: Split $A$ into $(Attr_1, .., Attr_n)$
2: $statement \leftarrow$ ( SELECT $Attr_1, .., Attr_n$ FROM $P$'s table GROUPBY $Attr_1, .., Attr_n$ HAVING $COUNT(*) > f$ AND $c$ )
3: $results[] \leftarrow executeQuery(statement)$
4: return results
```

not exception-based access entries. Given a more restrictive or totally different schema, the problem of separating violations from useful exceptions in an audit trail may require more sophisticated algorithms and even further research.

Algorithm 6. $Prune(Patterns, P_{PS}, V)$

```
Require: $\exists getCardinality(S)$ (returns the cardinality of a set S)
Require: $\exists getRange(P, V)$ (returns the range of the policy $P$ according to the policy
    vocabulary $V$)
Require: $\exists getComplement(S_x, S_y)$ (returns the 'set complement'of $S_x$ and $S_y$)
1: $range_x[] \leftarrow getRange(P_{PS}, V)$
2: $range_y[] \leftarrow getRange(Patterns, V)$
3: $usefulPatterns[] = getComplement(range_x, range_y)$
4: return $usefulPatterns$
```

Extract Patterns. In this step, the exceptions provided by the *Filter* phase, referred to as *Practice* in Algorithm 3, are analysed using a standard *data analytics* technique. The process is outlined in Algorithm 4.

To do the data analytics, a simple routine is called that takes a set of attributes, A, which is (a subset of) our audit schema, a minimum frequency, f, and a simple condition, c, translates it into a SQL statement and executes it on *Practice* to retrieve a list of entries that have occurred at least f and satisfy condition c (Algorithm 5). The technique finds the exact rules that have occurred more than f times. The data analysis routine has a well-defined interface that allows the *extractPatterns* algorithm to evolve and be easily customizable.

Prune. Not all the patterns produced from the extraction phase may be good candidates for inclusion into P_{PS}. As a first step in determining these useful patterns, we implement a prune mechanism, Algorithm 6, that removes the patterns that are already present in P_{PS}. This is where our implementation of prune ends, because we recognize that some patterns may represent behavior that needs to be stopped. This implies that human input is prudent at this stage to determine which patterns are actually good practice and which should be investigated or terminated.

5 Use Case Scenario

We will now illustrate the use of PRIMA in a realistic healthcare use case scenario. We will refer to the system for which the policies tied to the policy store and audit logs have already been described in Section 3.3.

We have already defined the basic audit trail schema as $\{$*(time,t_j), (op,X_j), (user,u_j), (data,d_j), (purpose,p_j), (authorized,a_j), (status,s_j)*$\}$. Building on the policy store P_{PS}, audit logs P_{AL} and policy vocabulary V used in Section 3.3, the audit trail generated by the system is shown in Table 1. Here we assume that the audit logs have been maintained for a period sufficient to be considered as the training period for this system and none of the exceptions reported in the logs are violations.

Invoking *ComputeCoverage(P_{PS}, P_{AL}, V)* on this snapshot of the audit logs reveals that the coverage has actually dropped to 30%. This is because the ratio of matching rules to total rules between P_{PS}, as per Figure 3, and P_{AL}, as per this snapshot, is now 3/10. In order to improve the coverage, we will run the *Refinement* algorithm. At line 1 of this algorithm, the *Filter(P_{AL})* function

Table 1. Audit trail, P_{AL}, for the system described in Figure 3

Time	**Op** (1:allow)	**User**	**Data** (Category)	**Purpose**	**Authorized** (Role)	**Status** (0:Exception)
t1	1	John	Prescription	Treatment	Nurse	1
t2	1	Tim	Referral	Treatment	Nurse	1
t3	1	Mark	Referral	Registration	Nurse	0
t4	1	Sarah	Psychiatry	Treatment	Doctor	0
t5	1	Bill	Address	Billing	Clerk	1
t6	1	Jason	Prescription	Billing	Clerk	0
t7	1	Mark	Referral	Registration	Nurse	0
t8	1	Tim	Referral	Registration	Nurse	0
t9	1	Bob	Referral	Registration	Nurse	0
t10	1	Mark	Referral	Registration	Nurse	0

filters out the log entries which are marked as non-exceptions, and therefore the *Practice* array now contains only the entries recorded at $t3$, $t4$ and $t6 - t10$.

The next step is to run data analytics to get the patterns that are candidate for inclusion in the policy. This is done at line 2 of Algorithm 2, when *extractPatterns*(P_{AL}, V) algorithm is called. As first steps in this algorithm, the relevant variables are set to enable data analysis ($A = \{data, purpose, authorized\}$, $f = 5$, $c =$"$COUNT(DISTINCT(User)) > 1$"). The output of the *dataAnalysis* (P_{AL}, A, f, c) routine returns those ($data, purpose, authorized$) tuples in P_{AL} that occur at least 5 times. In this instance, the pattern is $Referral : Registration : Nurse$, i.e. tuples t3 and t7-t10.

As the last step, in line 3 of the *Refinement* algorithm, *Prune*($Patterns, P_{PS}, V$) is called to obtain the useful patterns from the ones in *Patterns*. The prune algorithm works by taking the ranges of both P_{PS} and *Patterns* and then getting the 'set complement'of their intersection. This resulting set effectively contains those patterns that are not covered by existing rules in the policy store.

Thus, at the end of the *Refinement* algorithm, *Patterns* contains $Referral$: $Registration : Nurse$ which is recorded in entries at t3 and t7-t10. This reveals that a Nurse accesses the Referral data for a patient too frequently for Registration reasons using the exception mechanism. Assuming that this is not a negative trend, then it suggests that a rule should be included in the policy stating that Nurses may be allowed to access patient Referral data for Registration purposes.

We are cognizant that the criterion used for pattern extraction, such as the threshold frequency of rules and numbers of users involved, is clearly subjective and this scenario only serves to illustrate our approach and is not meant to be a definitive solution. The PRIMA systems will need to be configured and tuned as per the requirement specifications of the target environment. Secondly, simple data analytics techniques may not be sufficient in all cases. In order to enable a bit more sophisticated inference, we propose to leverage the frequent pattern mining algorithm [18] in our future work to detect correlations between attribute pairs that are not discovered by simple SQL queries.

6 Conclusion

In this paper, we formally introduced the problem of *policy coverage* in healthcare systems, which emerges from the over-reliance on the bypassing of security controls to access sensitive medical information, a phenomenon which is referred to in the medical community as *Break The Glass*. Our formalization is supported by PRIMA, a PRIvacy Management Architecture for healthcare systems, which addresses this problem of the circumvention of policy. PRIMA utilizes the actual practices of the organizations (embodied in the audit logs) to perform policy refinement. The system's advantages are that (i) it fits to the clinical workflow and does not require the workflow to fit to it, i.e. it does not impede the clinical workflow, (ii) it enables precise (or rather more realistic) definitions of purposes, criteria for exception-based accesses and categories of authorized users, and (iii) it enables improved privacy protection for the patient.

While emerging healthcare organizations leverage relational database systems, legacy systems employ hierarchical, XML-like structures. Thus, the natural evolution for PRIMA is to adapt the core concepts and technology to the tree-based structures.

References

1. Pear, R.: Warnings over privacy of us health network. New York Times (February 18, 2007)
2. Rostad, L., Edsburg, O.: A study of access control requirements for healthcare systems based on audit trails from access logs. In: Proc. of the 2006 Annual Computer Security Applications Conference, Miami Beach, FL, USA (December 2006)
3. Wong, R.: An overview of data protection laws around the world. http://pages.britishlibrary.net/rwong/dpa.html
4. Ministry of Internal Affairs, Communications Information, and Communications Policy. Personal data protection law. http://www.kantei.go.jp/jp/it/privacy/houseika/hourituan/index.html
5. Health insurance portability and accountability act, u.s. department of health and human services. http://www.hhs.gov/ocr/hipaa/
6. Office of the Privacy Commissioner of Canada. Personal information protection and electronic documents act. http://www.privcom.gc.ca/legislation/02_06_01_01_e.asp
7. Break-glass an approach to granting emergency access to healthcare systems. http://www.nema.org/prod/med/security/upload/Break-Glass-Emergency_Access_ to_Healthcare_Systems.pdf
8. United states presidential directive. http://www.himss.org/CPRIToolkit/html/4.11.html
9. Hand, D.J., Mannila, H., Smyth, P.: Principles of data mining (August 2001)
10. Agrawal, R., Kiernan, J., Shrikant, R., Xu, Y.: Hippocratic databases. In: Proc. of the 2002 Very Large Data Bases, Hong Kong, China (June 2002)
11. IBM. Ibm hippocratic database active enforcement (version 1.0): User's guide. http://www.almaden.ibm.com/cs/projects/iis/hdb/Publications/papers/HDBEnforcementUserGuide.pdf
12. IBM. Ibm hippocratic database compliance auditing (version 1.0): User's guide. http://www.almaden.ibm.com/cs/projects/iis/hdb/Publications/papers/HDBAuditingUserGuide.pdf
13. Blobel, B.: Authorisation and access control for electronic health record systems. International Journal of Medical Informatics 73(3) (2004)
14. Anderson, R.: A security policy model for clinical information systems. In: Proc. of the 1996 IEEE Symposium on Security and Privacy, Oakland, CA, USA (May 1996)

15. Bhatti, R., Moidu, K., Ghafoor, A.: Policy-basd security management for federated healthcare databases (or rhios). In: Proc. of the 2006 International Workshop on Healthcare Information and Knowledge Management, USA, November (2006)
16. Weaver, A.C., Dwyer III, S.J., Snyder, A.M.: Federated, secure trust networks for distributed healthcare it services. In: Proc. of the 2003 IEEE International Conference on Industrial Informatics, Alberta, Canada (August 2003)
17. Ihe patient care coordination technical framework: Basic patient privacy consents, supplement 2005-2006 (August 2006)
18. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. of the 1994 Very Large Data Bases, Santiago, Chile (September 1994)

Requirements of Secure Storage Systems for Healthcare Records

Ragib Hasan[1], Marianne Winslett[1], and Radu Sion[2]

[1] University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{rhasan,winslett}@cs.uiuc.edu
[2] Network Security and Applied Cryptography Lab
Stony Brook, NY 11794, USA
sion@cs.stonybrook.edu

Abstract. Recent compliance regulations are intended to foster and restore human trust in digital information records and, more broadly, in our businesses, hospitals, and educational enterprises. In the health sector, storage and management of electronic health records have become a vital issue. Specifically, with the passing of the Health Insurance Portability and Accountability Act (HIPAA), the security of medical records has come into focus. HIPAA and other regulations in the health sector require strict compliance with specific privacy and security requirements. Unfortunately, existing storage solutions do not live up to the task of ensuring compliance with mandated legislation. In this position paper, we discuss the main characteristics of the health sector record management regulations, and present a set of requirements for secure, trustworthy storage that complies with these regulations. We also briefly analyze existing storage models, and show that they are not suitable for meeting the requirements of health-care record storage.

1 Introduction

Accurate and detailed record-keeping, along with ensuring their privacy and authorized access, are integral parts of managing medical information. With the advent of electronic computing, medical records, like many other application domains, have depended heavily on computerized storage systems for storage and archival of health information.

However, in the digital realm, the adversaries and attackers are quite different than the physical world – digitally stored information can be copied verbatim, and records may be exposed to a wide variety of adversaries. To protect privacy and security of such electronic medical records, many countries worldwide have enacted consumer protection and privacy laws. These laws have strict guidelines and requirements for regulation of medical record management.

Unfortunately, existing storage architectures are not capable of providing the strong security and privacy guarantees mandated by the laws associated with this new digital information domain. For example, several regulations require mandatory record retention (with data integrity) for periods of up to 30 years. But storing such records for a long time would require inevitable change of storage hardware and/or storage format. The resulting migration to new servers must be trustworthy, and verifiable. Similarly, if a medical record needs to be removed after the mandated retention period, the storage system must guarantee its secure deletion. Such features are not available in most of the current storage architectures for medical records.

In this paper, we look into HIPAA and several other regulations on protection of medical records, and discuss the security and privacy requirements that such regulations impose on record management. From this discussion, we derive a set of common requirements for electronic health-care record storage systems. Finally, we briefly look into several storage architectures, and show the limitations of current architectures in meeting all the requirements. The contribution of this paper is to map out the open research problems in the area, and to direct future research endeavors for secure storage of health-care records.

2 Health Care Regulations

Management of health information has become an important and regulated area in most countries. In the following, we briefly discuss several of these laws from different countries, and outline their essential common mandated features.

2.1 HIPAA

The Health Insurance Portability and Accountability Act of 1996, commonly known as HIPAA [3,7], is an attempt to update the health sector and insurance record keeping in order to bring more accountability and better protection of consumer rights. Besides regulating the insurance industry, one of HIPAA's significant effect is to mandate the confidentiality and integrity of medical information.

HIPAA is divided into two titles. Title I regulates health insurance coverage. Title II discusses digital health care records, their security, privacy, as well as other facets of their management. The following main security and privacy requirements are mandated by HIPAA:

– **Privacy and Data Confidentiality.** The privacy rule of HIPAA requires organization to ensure that they have taken reasonable steps to ensure the confidentiality of health care records and communication with individuals. Individuals have the right to request correction of health care records.
– **Security.** Organizations outsourcing some of their record management tasks must ensure that the third-parties also comply with HIPAA. Each organization must have established *internal audit* procedures for medical records. All records must be disposed of in a trustworthy manner at the end of their *retention* period. Access to hardware and software should be limited to properly authorized individuals. Data *integrity* must be ensured by means of checksums, message authentication, or digital signatures. Each entity is responsible for ensuring that data within its systems have not been erased or tampered with.

Specifically, the General Rule (Section 164.306) requires entities to:

- Ensure the confidentiality, integrity, and availability of all electronic protected health information (EPHI) the covered entity creates, receives, maintains, or transmits;
- Protect against any reasonably anticipated threats or hazards to the security or integrity of such information;
- Protect against any reasonably anticipated uses or disclosures of such information that are not permitted or required by the Privacy Rule; and
- Ensure compliance by its workforce.

Additionally, Section 164.310 of HIPAA mandates storage media disposal, media-reuse, accountability, and data backup/storage for medical records. It requires the following:

– **Disposal.** 164.310(d)(2)(i) requires that covered entities must have policies and procedures that handle the final disposition of electronic health information records, and the media or hardware on which the records are stored.
– **Media re-use.** 164.310(d)(2)(ii) states that covered entities must implement "procedures for removal of electronic protected health information from electronic media before the media are made available for re-use."
– **Accountability.** 164.310(d)(2)(iii) states this: the covered entity must "Maintain a record of the movements of hardware and electronic media and any person responsible therefore." In other words, organizations must *log all data migration and data provenance information.*
– **Backup and Storage.** Finally, 164.310(d)(2)(iv) mandates that a covered entity "must create a retrievable, exact copy of electronic protected health information, when needed, before movement of equipment."

2.2 Occupational Safety and Health Administration Regulation

In the United States, the Occupational Safety and Health Administration regulation (Standards - 29 CFR) "Access to employee exposure and medical records.-1910.1020" [11], controls the management of medical records for employees, and all exposure records. Section 1910.1020(d)(1)(ii) requires that "Each employee exposure record shall be preserved and maintained for at least thirty (30) years". It also requires all employee medical records to be kept for at least 30 years. For businesses changing ownership, it must ensure the transfer of the records to the new owner.

2.3 EU Directives

In Europe, the Directive 95/46/EC of the European Union on the protection of personal data, provides privacy and security guarantees for personal information, including health care records [4]. In particular, Article 6 of the directive requires accuracy guarantees of personal records, and guaranteed disposal after the retention period. Article 17 requires measures for ensuring the confidentiality and availability of records. In addition, most countries in Europe have their own data protection laws. For example, in the United Kingdom, the Data Protection Act of 1998 [2] regulates, among other information, personal health-care records. It requires mandatory disposal of electronic records after retention period, accuracy of information, logging any changes, and strict confidentiality.

3 Requirements

As seen in the previous section, a set of relatively consistent, broadly mandated assurances can be found in a multitude of regulations. In the following we discuss the main requirements that storage systems would need to adhere to, for compliance purposes, including data confidentiality, records integrity and availability, as well as secure retention, deletion and migration mechanisms.

Confidentiality and Access Control. As health-care records contain sensitive information, the storage systems must ensure their confidentiality. Moreover, only authorized personnel should have access to confidential medical records. Consequently, to ensure confidentiality, storage systems must deploy strong encryption in both the actual storage and the data pathways leading to and out. Moreover, in the case of storage media re-use or disposal, the confidentiality of records previously stored in such media should be ensured.

Integrity. The storage system must ensure the integrity of medical records. In particular, it must ensure the integrity of medical records even in the case of malicious insiders. The security mechanisms must identify any tampering of information.

Availability and Performance. The health-care records must be accessible in a timely manner. Medical records are frequently expanded, and patients may also ask for correction of records. Hence, appropriate storage models should be used to allow both performance, security, and mutability.

Timely access to medical records would require indexing techniques. However, regular indexing schemes such as keyword index can breach privacy as the mere existence of a word in a document can leak information [9]. For example, if the keyword “Cancer” is present in a medical, then an adversary can assume that the patient might have Cancer. So, the index itself must be trustworthy, and confidential.

Logging, Audit Trails, and Provenance. All access to the storage system should be logged in a trustworthy manner. HIPAA mandates recording all medical record access information. Many of the regulations require extensive logging to record the movement of records between systems, and the access and modification history. Consequently, the storage system must provide verifiable audit trails and the maintenance of provenance information on the chain of records custody.

Support for Long Retention and Secure Migration. Many of the regulations require long retention periods for certain types of health-care records. The storage system must be capable of providing long term retention guarantees.

Since it is conceivable that the failure of storage servers, as well as obsolescence of technology and formats will require migration of records, the storage system must provide trustworthy and verifiable migration mechanisms.

Backup. There must exist strong backup and restore operations. The backup copies should be located in a separate off-site location to ensure survival in case of fire or natural disasters.

Cost. The storage system must also be cost effective, possibly using cheap off-the-shelf hardware. Compliance with HIPAA and other regulations have significant management overhead. The cost of training personnel is also another factor. So the storage system should notbe cost-prohibitive. Media used by the storage system should be cheap.

4 Limitations of Existing Storage Models

We now discuss the suitability and limitations of existing storage models with respect to the above requirements. In particular, we look at relational databases, object-storage systems, and compliance WORM storage.

Commercial solutions for HIPAA compliant storage tends to focus on using strong encryption to provide security for electronic records [13]. Unfortunately, however, such schemes do not protect against malicious insiders. Moreover, such encryption based solutions do not account for maintaining provenance information.

Most of the early storage systems for electronic records involved relational databases. However, securing relational databases to the extent of compliance with the requirements described in section 3 is difficult. Relational databases are geared more towards performance rather than security. Specifically efficiently performing queries on encrypted data in the presence of malicious insiders as well as guaranteeing secure record retention are significant open problems to consider.

A promising alternative is IBM's Hippocratic Database Technology [6], which aims at providing regulatory compliance with data protection laws. It provides fine-grained access control by transparently rewriting user queries and enforcing various access and disclosure policies. Hippocratic databases also provide compliance auditing, in which database access information is logged for future forensic analysis in case of a privacy breach. However, without underlying security support, just defining semantics and enforcing them in a software query processor still leaves things vulnerable to insider attacks with direct disk access.

In object based storage systems, usually document content hashes are used as object IDs to locate documents [8]. This renders such mechanisms suitable for efficient storage of read-only content and read operations are efficient and optimized. Moreover, information integrity can be easily assured. However, appends and writes in the presence of malicious adversaries are difficult to achieve in object storage, and likely slow in performance.

The most promising technology for secure storage of health records is compliance WORM storage [5,9,10]. In such systems, records are kept in write-once, read-many times storage media. The media can be optical, or magnetic. Trustworthy indexing mechanisms [9] can ensure fast retrieval of data, as well as ensuring privacy and integrity of the index. Trustworthy migration [10] can ensure guaranteed and verifiable transfer of records among systems. Trustworthy deletion mechanisms can ensure complete removal of expired records. However, compliance WORM storage is mainly suitable for records that do not require corrections. Since medical records are expected to be corrected, and individuals have the right to request such corrections to their medical records, allowing corrections is an important feature. Currently, trustworthy WORM storage systems do not support such corrections.

Ultimately, the trade-off between security and performance makes it difficult to use existing secure storage systems. Most of the existing systems are geared towards read-only settings, optimizing read operations via smart indexing and caching. However, to support efficient and trustworthy write operations, data retention, secure deletion, migration, such systems simply do not live up to the requirements.

Moreover, an additional missing feature in all these systems is storage of provenance information [1,12]. Since access to storage records must be recorded for later audits, it is critical to record such information. With migration of records between different systems, it is important to ensure a proper chain of custody for the ownership and transfer of records. However, current storage systems do not implement trustworthy provenance, and therefore, cannot fulfill this requirement of health-care record storage.

5 Conclusion

In this paper, we explored major health care regulation acts and discussed their impact on the requirements for associated storage support systems. We showed that unfortunately, existing systems and data models fall short of the resulting desiderata. We thus believe it is important to explore novel avenues and solutions in this area that would possibly combine existing functionality creating a hybrid model suited for trustworthy regulatory-compliant health-care record storage. Additionally, it is important to explore and consider the impact of the additional costs and overhead burdens such mechanisms would put onto their users and the healthcare system in general. Ultimately, as increasing amounts of health information are created and stored digitally, we believe compliance storage to be a vital tool in providing trust and privacy assurances.

Acknowledgments

The authors would like to thank the anonymous reviewers for their suggestions. This research was supported by the NSF under grants IIS-0331707, 0331690, CNS-0325951, and CNS-0524695. Radu Sion was in part supported by the NSF through awards CT CNS 0627554, CRI CNS 0708025 and CT CNS 0716608.

Sion would also like to thank Motorola Labs, IBM Research, CEWIT and the Stony Brook VP Office for Research for their support.

References

1. Braun, U., Garfinkel, S., Holland, D., Muniswamy-Reddy, K.-K., Seltzer, M.: Issues in automatic provenance collection. In: Proceedings of the International Provenance and Annotation Workshop, pp. 171–183 (2006)
2. British Parliament. Data protection act of 1998 (1998), Online at: http://www.staffs.ac.uk/legal/privacy/dp10rules/index.php
3. Center for Medicare & Medicaid Services. The Health Insurance Portability and Accountability Act of 1996 (HIPAA) (1996), Online at: http://www.cms.hhs.gov/hipaa/
4. European Parliament. Legislative documents (2006), Online at: http://ec.europa.eu/justice_home/fsj/privacy/law/index_en.htm
5. Hsu, W., Ong, S.: WORM Storage is not Enough. IBM Systems Journal (April 2007)
6. Johnson, C., Grandison, T.: Compliance with data protection laws using hippocratic database active enforcement and auditing. IBM Systems Journal 46(2) (2007)
7. Lawson, N., Orr, J., Klar, D.: The HIPAA privacy rule: An overview of compliance initiatives and requirements. Defense Cousel Journal 70, 127–149 (2003)
8. Mesnier, M., Ganger, G., Riedel, E.: Object-based storage: pushing more functionality into storage. IEEE Potentials 24(2), 31–34 (2005)
9. Mitra, S., Hsu, W., Winslett, M.: Trustworthy keyword search for regulatory-compliant record retention. In: Proceedings of the 32nd International Conference on Very Large Data Bases, pp. 1001–1012. ACM, New York (2006)
10. Mitra, S., Winslett, M.: Secure deletion from inverted indexes on compliance storage. In: StorageSS '06: Proceedings of the Second ACM Workshop on Storage Security and Survivability, pp. 67–72. ACM Press, New York (2006)
11. Occupational Safety and Health Administration. Access to employee exposure and medical records. - 1020 regulations (standards - 29 cfr) (1910), Online at: http://www.osha.gov/pls/oshaweb/owadisp.show_document?p_table=STANDARDS &p_id=10027
12. Simmhan, Y., Plale, B., Gannon, D.: A survey of data provenance in e-science. SIGMOD Rec. 34(3), 31–36 (2005)
13. Smart Card Alliance. HIPAA compliance and smart cards: Solutions to privacy and security requirements (September 2003), Online at: http://www.datakey.com/resources/HIPAA_Compliance_and_Smart_Cards_FINAL.pdf

An Intrusion Detection System for Detecting Phishing Attacks

Hasika Pamunuwa[1], Duminda Wijesekera[1], and Csilla Farkas[2]

[1] George Mason University, 4400 University Drive, Fairfax, VA, USA, 22030
[2] University of South Carolina, Columbia, SC 29208
hpamunuw@gmu.edu, dwijesek@gmu.edu, farkas@cse.sc.edu

Abstract. In this paper we present a hybrid system to protect private data from phishing attacks. Our solution uses intrusion detection methods to identify potential phishing emails then relies on web crawlers to validate or reject this suspicion. We also use external information coming through RDF Site Summaries (RSS) alerts about potential phishing sites. Our two-layered phishing detection system reside on the server of the organization, thus it is not vulnerable to blocking attacks targeting web browsers.

1 Introduction

One of the emerging serious threats against personal data security is phishing. Phishing attacks are often performed by sending out emails that seem to originate from a trusted party. The objective is to deceive the recipient to release sensitive information such as usernames, passwords, banking details, or credentials [8]. The message attempts to convince the receiver that it is to his/her benefit to enter the requested information at the indicated web site. After obtaining this information, the phishers may use it for fraudulent activities, like identity theft. Current solutions to counteract phishing attacks are based on identifying potential phishing sites using browser plug-ins. However, techniques to block such identification have emerged. Moreover, identification of phishing sites is not trivial, resulting in missed detection of attacks.

In this paper, we propose the use of an organization-wide and server-based intrusion detection system (IDS) for identifying phishing email and evaluate the sites they refer to. Our system, shown in Figure 2, consists of two major components: (a) a system-wide data capturing facility that categorizes some of the incoming e-mails as potential phishing activities, and (2) a secondary validation/refutation system that crawls the advertised sites to decide whether they are phishing sites.

More specifically, the contributions of this work are four fold. First, we propose an organizational level architecture to detect phishing activities in incoming traffic. By doing so, phishing traffic can be filtered at the perimeter of the organization, thus limiting the dependence on individual users' security practices and browser-based phishing detection methods. Second, we use intrusion detection system to detect potential phishing e-mails. This method ensures that all incoming traffic is evaluated. Third, upon the detection of suspicious e-mails, our crawlers visit the web sites indicated by the e-mail and recursively follow all links from the potentially phishing sites. This step is motivated by the observation that most phishing sites eventually visit the site that they impersonate. We use this observation in the final validation step, leading to the fourth contribution. That is, we recursively traverse all web sites referred to in the site under investigation. When this recursion terminates, our algorithm strips off the graphics of the original and potentially phishing sites and assigns a score for the differences of the web sites.

The rest of the paper is as follows. Section 2 describes related work. Section 3 describes the proposed system. Section 4 describes the results of our experiments. Section 5 describes the experimental results and Section 5 concludes the paper.

2 Phases of Phishing Attacks and Current Solutions

In this section we give an overview of the different phases of a phishing attack and the currently available phishing prevention methods.

2.1 Phases of Phishing

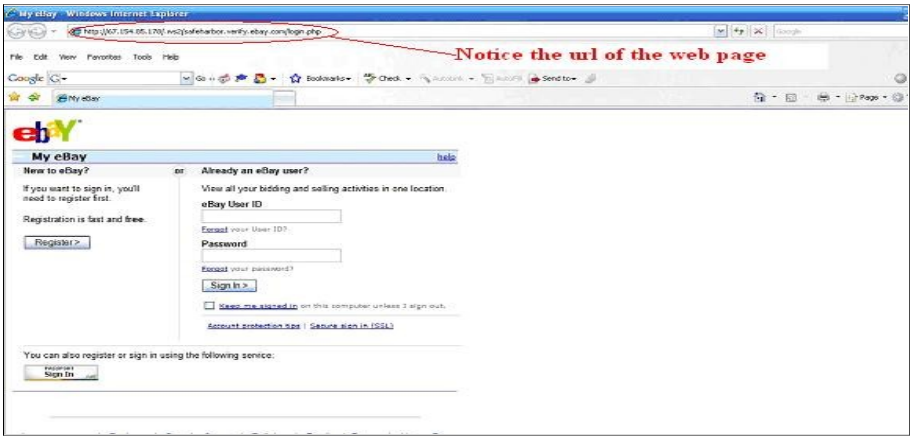
Our method to detect phishing attacks exploits the phases of a phishing attack [2]: (A) Planning, (B) Setup (C) Attack (D) Collect information (E) Commit fraud (F) Post-attack clean-up.

A. Planning: The first step of the planning activity is to select an institution to be phished, such as a bank or credit card service. These institutions generally provide online services, and require specific information from their customers. Common examples are credit card information and credentials to e-commerce web sites. For example, a phisher seeking credit card information would send e-mail requesting confirmation of credit card details. The sites pointed to in phishing email are impersonations hosted on a compromised machine. They contain HTML forms to be filled out to confirm their records.

The second step of the planning activity is to select the method of information extraction. Common examples are putting an HTML link in an e-mail message to register within a domain with a sub domain that closely matches the impersonated institutions URL. An example phishing on eBay sign-in page may point to the web site http://signin.ebay.com.example.com where the actual eBay sign-in page is https://signin.ebay.com. The objective of this stage is to design a URL that will be difficult for a reader to distinguish from that of the legitimate institution. In particular, novice users may fail to recognize the fine differences in the addresses, and reveal their information.

Figure 1 show an instance of a phishing site that was impersonating ebay.com's sign-in page. The site is hosted at http:// 67.154.85.178/ .ws2/ safeharbor.verify.ebay.com/login.php and uses the URL of a compromised machine. Note the hierarchy of the location; it ends with ebay.com in its path which gives the impression to the novice user that this is the original ebay.com's sign in page.

The third step of the planning stage is to find a compromised machine that could host the phishing web-site. In order to ensure that the phishing site does not arouse victim user's curiosity or doubt, they are designed to closely resemble the style sheets and the structures of the victim institution. Detecting web-page structure is extensively discussed in [3]. We use their method in this work for the detection of phishing sites. Wenyin et al. [9] describe three metrics to measure the visual similarity of two sites; block level similarity, layout similarity and overall styling similarity. An additional observation of us is the string similarity between the phishing site and the phished site.


My eBay - Windows Internet Explorer
http://67.154.85.170/.ws2/safeharbor.verify.ebay.com/login.php
Notice the url of the web page
File Edit View Favorites Tools Help
Bookmarks Check
My eBay
My eBay
help
New to eBay?
or
Already an eBay user?
eBay User ID
Password
Register >
Sign In >
Sign In

If you want to sign in, you'll need to register first.

Registration is fast and free.

View all your bidding and selling activities in one location.

Forgot your User ID?

Forgot your password?

Keep me signed in on this computer unless I sign out.

Account protection tips | Secure sign in (SSL)

You can also register or sign in using the following service:

Fig. 1. A Phishing web site for eBay.com

Web pages can be designed in HTML using numerous ways, such as using table structures using Cascading Style Sheets based structures and structural changes defined using JavaScripts.

Hence it is not always true that phished site and the phishing sites should used the same method of structural specification. But a phisher cannot change much is the contents of the page to remain authentic-looking. Consequently, we extract the content from a site and strip off all the HTML tags obtain all strings that belong to a page and use the remaining text content to distinguish between the phishing and phished sites.

B. Setup: Phishers setup all the resources used in the rest of the process during this stage. First they design and setup the HTML structure of the target site. As stated, the styling of a phishing site is similar to that of the victim institution. A phishing page always contains a *HTML form page* for the victim user to post information to the fraudulent web server. This form may be an exact copy of the form in the phished site or a slightly modification. Because the primary objective is to obtain all information entered in the form, phishers may use server side scripting to post the information on a specific location. If e-mail is used as the primary delivery mechanism of the captured information, there would be a mail server residing on the same compromised machine. Some possible delivery mechanisms are:

1. Store in a database or a flat file
2. Send as text message to a mobile device
3. Post to a message forum

Because of the short life-times of phishing sites phishers are known to thoroughly test their delivery mechanisms.

C. Attack: During this stage the phisher lures victimized customers to a phishing URL. The most commonly way of contacting victims is using e-mail [1]. Other methods are to post messages with links on public chat rooms, via message boards, via newsgroups, and subscribed RSS feeds.

D. Collection: During this stage, the phisher collects the responses from victims. Phishing sites contain an HTML form with an action method that sends an e-mail, stores or alerts a mobile device on obtaining sensitive data.

E. Fraudulent Use of Phished Information: During this period, phished information is used for fraudulent activity. Most common activities are:

1. **Using collected credentials:** If captured information contained credit card or any financial source, they are used to purchase or pay for services provided on the Internet.
2. **Credentials used to second stage attacks:** If the captured information had credentials for an access controlled site, use them to gain entry.
3. **False registration:** Use personal identity attributes to impersonate and obtain online privileges

F. Post Attack Activity: Once the attacker reaches their goal of victim they would deliberately shut down the server to evade capture by authorities. They would switch back and forth as to give an impression that the server is no longer available and this could help them to deceive any monitors on these compromised machines.

2.2 Phishing Prevention

Current solutions use strong spam filters to isolate phishing solicitations or capture phishing sites at the browser. For example, Zhang et al. [8] propose a solution using *term frequencies* and *inverse document frequencies* between a potentially phished and phishing sites to classify the latter as a phishing site.

Fette et al. [4] proposes PILFER, an e-mail filter that uses a classifier based on 10 features relevant to phishing activity. This solution has been able to classify phishing email with a true positive rate of 92% and a false positive rate of 0.1%. The 10 features used for classifying are: (1) IP-based URLs. (2)Age of links (domain names) – this is calculate by querying *whois* data. (3) Non-matching URLs – these are basically masked HTML links. (4) "Here" links to non modal domain. (5) HTML emails. (6) Number of links. (7) Number of Domains. (8) Number of dots – dots with in a URL. (9) Contains JavaScript. (10) Untrained SpamAssasin Output.

This model uses a sample training dataset. A freely available SVM library is used as the classifier. The system has also been tested with other approaches, such as Bayesian classifying, decision trees and rule based approaches. The dataset used for this experiment uses 6950 non phishing emails and 870 phishing emails.

SpamAssassin [14] is another tool that recognizes spam containing phishing email. [4]state that SpamAssassin has a false negative of 15% for spam e-mails, and performs worst when tested with 10 fold cross validation. SpamAssassin uses a wide range of heuristic tests on mail headers in order to identify spam, and can be customized. For algorithms, it uses text analysis, Bayesian filtering, *DNS blocklists*, a collaborative filtering database, and a *Stochastic Gradient Descent* method in training a neural network. This is used for its scoring based on perception that uses a single perception with a logsig activation function that maps the weights to SpamAssassin's score space. SpamAssassin does not delete email from mail boxes, but it can route classified e-mail to mail boxes or folders.

Chou et al. [5] proposes a browser-based plug-in, *SpoofGuard*, that monitors users internet activities and warns if the tool classifies a visiting web-site as a phishing page. SpoofGuard uses the observation that a page is loaded from an e-mail message and whether the URL was visited before. The authors propose the use of the following properties: (1) Logos – use of images. (2) Suspicious URLs – urls that contains IP address or higher length urls. (3) User Input – pages that has form input. (4) Short lived – the spoof sites are shut down with in 2 – 3 days. (5) Copies – similar contents. (6) Sloppiness or lack of familiarity with English – misspellings and grammar errors. (7) HTTPS is uncommon – do not user https with secure sockets layer.

SpoofGuard uses 3 methods to determine impersonation: (1) a stateless method that determines whether a downloaded page is suspicious, (2) a stateful method that evaluates a downloaded page in light of previous user activity, and (3) a method that evaluates outgoing post data. SpoofGuard uses a standard aggregate function to calculate the *total spoof score* (TSS) computed as:

$$\mathrm{TSS(page)} = \Sigma_1^{n} w_i P_i + \Sigma_{1,1}^{n,n} w_{i,j} P_i P_j + \Sigma_{1,1,1}^{n,n,n} w_{i,j,k} P_i P_j P_k \ldots$$

For a given downloaded web page and a browser state TSS produce a number P_i within [0,1] where 1 indicates a page more likely to be a spoof page. The w_i's are preset weight to minimize false positives. SpoofGuard has a configuration pop-up screen that requires a user defined spoof rating threshold. This allows setting independent weights and security levels for the domain name, url, link, password and image checks. The user interface alerts suspicious sites with a traffic light symbol lighting for the degree of the probable spoof activity. The information which was based for classifying is available for the user.

Even though a link from an e-mail is a good method for phishing detection, a user clearing the browser history could result in many false positives. Sensitivity decreasing on this system would result in false negatives while increasing would result in false positives. Also as shown in [5] client side detection systems prevention a single user using one web browser. Conversely, intrusion detection systems, if usable could detect and remove phishing sites in large scale.

Most known systems use some type of trapping mechanism that alerts for potential spam. They setup a *sensitivity level*, where higher values generate alerts for slightest probable spam and lower levels generates alerts for most probable spam. Firstly, these tools have false positives, and secondly, they could be disabled by using a Trojan, thereby opening the browser for phishing.

HoneyTank [7] collects Spam using a honeynet and automatically generates a pattern. The pattern is to be used by a network based intrusion detection systems.

Thus their system is similar to ours, but does not crawl or use Snort to classify potential phish. A HoneyTank is a workstation receiving TCP segments sent to unallocated IP addresses and replying to those segments to emulate real end systems that supports TCP services. They use Advanced Sequential Analyzer on Unix (ASAX) as the intrusion detection system. ASAX is a generic system that analyzes sequential files like security audits trails. It is composed with three parts which are analyzer, rule declarations, and format adaptor. The analyzer receives the input from the format adaptor and analyzes according to the declared rules.

Cordero et al. [10] use rendered images as a basis to relate a potentially phishing site to a phished site. They propose a server-based solution using web pages rendered as images, where a whole page rendered as an image is used for classification as a phishing site, using the *safari* [15] browser based parser. This method uses the size of the rendered image size. This prototype is developed with safari HTML rendering engine *Cocoa* [16], *GNU Octave* [17] for data processing, *Image magick* [18] for image processing, python to hold all the processes together and the R statistical platform [20] for classification. But the rendered size of a web page differs due to the machine's screen resolution, and in particular, a lower screen resolution would render a web page out of the active viewing area. It is still questionable as to how well this system scales. Although this is a server based approach that attempts to identify phishing hosts before email containing their images are delivered to a client browser, its continuous monitoring of selected target sites may not scale well.

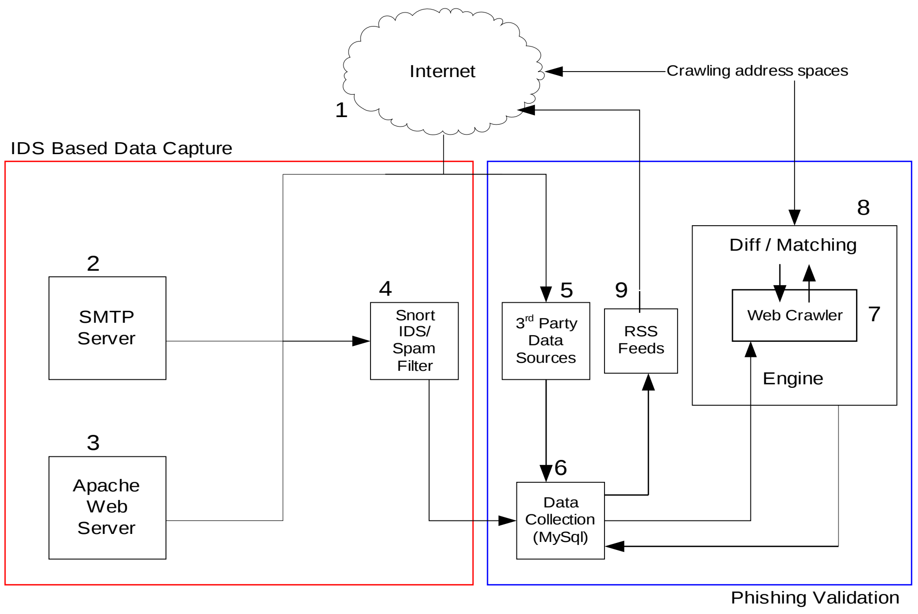


Fig. 2. The system architecture

3 Proposed Solution

We propose using an organization-wide and server-based intrusion detection system (IDS) to identifying phishing email and recursively crawl the identified sites. Our system, shown in Figure 2, consists of two major components: (a) a system-wide data capturing facility with a well-advertised mail server that seeks mail (with the hope of collecting phishing solicitations) and categorizes some of them as potential phish, and (2) a secondary validation or refutation component that crawls the potentially phishing sites. Organization-wide solutions allow the security officer to uniformly enforce security requirements, reducing the risk of violations due to lack of safeguards and updates.

3.1 Phishing Recognition IDS

The first component of our system consists of a SMTP server (#2 in Figure 2), an Apache web server (#3 in Figure 2), and a Spam filter (#4 in Figure 2). Our current implementation uses Snort for traffic filtering. The IDS scanner scans all received mail messages and flags some of them as potential phish, and enters them into a database (#6 in Figure 2). Emails that contain embedded HTML are classified as potentially phishing requests as in [4]. We use MySQL to store information about potential phishing messages, where database entries consist of the e-mail contents, time of receipt, and the originating IP address. A daemon process that runs on the same machine scans all URLs listed on flagged e-mail, strips off their HTML tags and identifies if they have been observed earlier. If the sites have not been evaluated previously, the phishing validation component tries to determine whether they are phishing sites. We use well documented characteristics of phishing e-mails to detect them. Our Snort filtering flag email as potential phish if they have some of the following characteristics:

1. HTML encoded in e-mail.
2. Any URL including IP addresses.
3. URLs that has been masked with HTML to a different address
4. Cross site images. They are scanned for further analysis.
5. All images that are not categorized as cross site reference are hashed using the MD5 algorithm for further analysis on the monitoring sites for phishing activity.

Although our simple Snort-based phishing IDS reorganization algorithm is quite primitive, our architecture has several advantages. First, the recognition of a phishing e-mail is independent of an individual user's settings and web browser based policies. Second, because it is server-based, it allows uniform security enforcement within an organization and reduces the efforts of performing updates. Third, our solution is not vulnerable to attacks that bypass client-browser based defenses. Fourth, our system is constructed so that the Snort based filtering algorithm can replaced by a more sophisticated versions, which constitute one aspect of our ongoing work.

3.2 Validating Phishing Sites

In our present implementation, the validation component uses two primary inputs. The first input contains the updated database entries. The second input contains the updates received from external trusted entities. These updates, emitted continuously, notifies about phishing sites using *Really Simple Syndication / Rich Site Summary or RDF Site Summary (RSS)* feeds [11], (#5 in Figure 2). We use a separate real-time engine to validate (or refute) the phishing email collected in the database by our first component (#8 in Figure 2). This engine extracts the web sites from potential phishing email, and uses web crawlers to visit these sites. The crawlers recursively follow links of the potentially phishing sites. All HTML pages from all visited locations are downloaded and evaluated by our algorithm. Our aim is to differentiate them from known sites that have been targeted for impersonation. Our algorithm starts with the list published by the *Anti-Phishing Working Group* [1] and extends it locally as our categorization grows. The high-level description of our algorithms is given in Table 1.

Table 1. High-level descriptions of algorithms used inside the matching algorithm

```
procedure
createLegitimateIndex(read links
from file)
{
        for each legitimate links
        {
                call nutch(link);
        }
        saveIndex();
        return indexPath();
}
```

```
procedure confirmPhishing()
{
   thread1:
   indexPath = creatLegitimateIndex(local file with links);

   thread2:
   if (linkAlert)
   {
       terms = crawlURL(url);
       documentID = matchTerms(terms, indexPath);
       probableHost = documentID.getHost();
   }
   return probableHost;
}
```

```
function crawlURL(url)
{
  create new http connection to url;
  parse HTML for each line;
  create terms from "<title>"
  or "<head>" tags
  save document content;
  for(ech links found within HTML)
  {
    //recursive call for further urls
found within HTML
    crawlURL(link);
  }
  release connection;
  return terms;
}
```

```
function matchTerms(String terms,legitimate index
path)
{
   extract first 10 lines;
   for (each line)
   {
        queryIndex(line,path);
        score[i] = doc.getLuceneScore;
        url = doc.get("url");
        if(score[i]>score[i-1])
        {
               if(url.equals(preUrl)
                     aggScore += score[i];
                 documentID = score[i].getSearchResult();
        }
    }
    return documentID;
}
```

As Table 1 show, the procedure creatLegitimateIndex(file) creates the legitimate search indexes to match the probably phishing terms. This procedure is initially set with a static e-commerce URLs that are marked from antiphishing work group as sites which were mostly phished in the recent past. This procedure crawls with Lucene

Nutch [21] and saves all the document contents as an index. The confirmPhishing() procedure takes a new row from the snort alert database table as input and it crawls on that URL and parse all its contents to a text file. This procedure first calls the procedure crawlURL(url) that takes a uniform resource locater URL as input and return an array of string string* and an array of links link* as outputs in one local buffer. The procedure confirmPhishing() then calls the function matchTerms(indexPath) with the legitimate index path to search for the closest term. matchTerm(terms,index) method calls a Lucene [21] search query on the legitimate search index with the terms derived from the crawl on the probable phishing URL.

The function matchTerm(terms,index) takes two inputs: an array of strings that do not contain URLs and an index path. It traverses the passed terms in order of the lines returned from the crawl. We search our index for each line and get the match score and count it to get the aggregate of the highest possible URL. We take the top four scoring URLs to analyze the matches.

The rationale behind this step is that most phished web sites finally point to the web site that it impersonates and therefore an automated recursive descent through its links using an automated crawler would find it. Finally, as shown in block #9 of Figure 2, we feed back our findings as an RSS feed to whoever wants to get our alerts.

4 Experimental Results

We have implemented the system as described in Section 3. For an experimental run, we collected a data set consisting of phishing sites as described in an anti phishing site and used them with our implementation. The reason for doing so was to determine the accuracy of detection, which is an important parameter of any IDS system. The outcome we obtained was analyzed along two dimensions: (a) accuracy and (b) timeliness, described respectively as follows.

Accuracy: We compared the detected phishing sites against the sites that are being impersonated. To do so, we indexed the HTML String content of all the legitimate web sites using an open source Web Crawler [21]. We crawled the web-sites which were flagged as phish by our snort sensors and captured their HTML string content. Then we queried our legitimate search indexes with the content we got as the result of our crawl on the phishing sites. Our objective of finding the closest matches against the legitimate index was achieved by aggregating scores from lucene queries.

1. Get the logs and look for false positives on matches
2. Graph for correctness of matching instances
3. Graph on false positives that were marked as matching instances
4. Scoring analysis, look for stats on sores – deviation curve of the scores
5. Accuracy through searching

The experiment was done with Snort capturing 313 URLs with 150 phishing pages and an additional 163 non-phishing HTML embedded e-mails. The contents stored in the final results files are:

1. Total returned lines URL1 – snort based phishing URL
2. Total returned lines URL2 – probable attacked URL
3. Start time (in milliseconds) when Snort first issued its alert to the database.
4. End time (in milliseconds) when we confirmed if the alert was correct or incorrect.

The total number of URLs that returned at least one matching lines were 100. The statistics of the number of lines matched are given in Table 2, and the individual scores of the matches are graphed in Figure 4, where the horizontal axis given the entry number in the database table and the vertical axis is the percentage of matching lines.

Table 2. Statistics

	Deviation on returned			
	Score 1	**Score 2**	**Score 3**	**Average**
Standard Deviation	0.14	0.08	0.05	0.06
Mean	0.36	0.26	0.21	0.28

Timing: We have collected the following times:

1. The time it takes to capture probable links upon a receipt of spam e-mail
2. Time it takes to feed to crawlers
3. Crawler Response time
4. Time to search through for activity
5. The time taken from beginning of the activity.

The graph in Figure 3 shows the time taken to crawl 181 pages, with an average of 7.4 seconds.

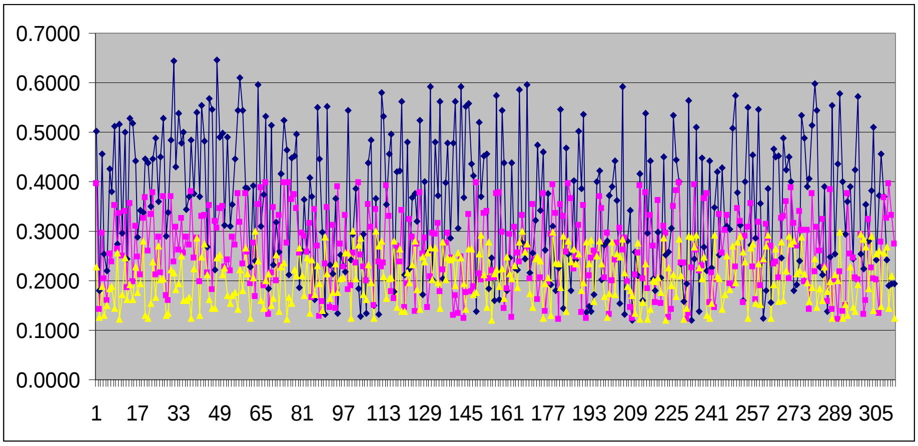


Fig. 3. Accuracy Scores

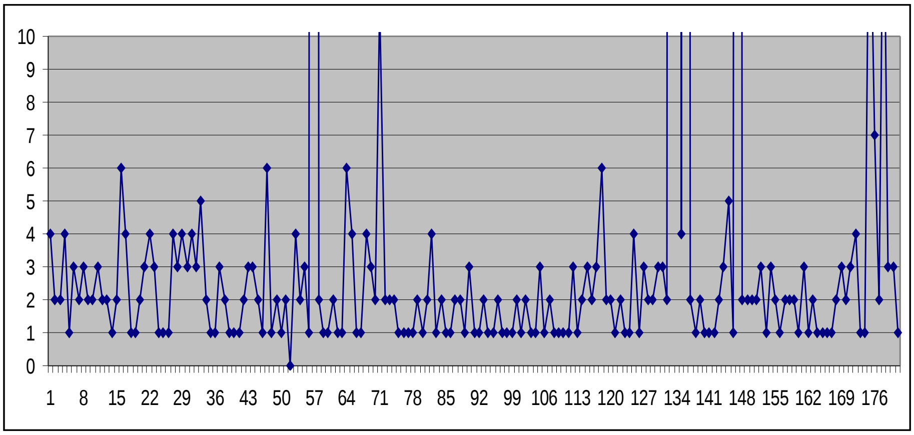


Fig. 4. Time to Crawl

The results were based upon the scores retrieved from the Lucene API. Figure 3 shows 3 scorings based on the best matches derived from the search queries on the index. The dark lines (in Fig 3) gave the best aggregate scores for matches, and the medium dark series providing the 2nd best and the lightest showing the 3rd. Our statistics shows the average scores given that the best matches for all 3 series. Figure 4 presents the time between DNS resolution and string extraction that includes crawling delays. Five data points exceeding the graph are dues to DNS resolution failures or sites being down.

5 Conclusions

Phishing is becoming a popular form of fraud on the Internet resulting in disclosure of personal data. Reacting to these attacks, the Internet community has responded by allowing the browsers to setup many methods to detect and warn of potential phishing attacks. Although successful, these solutions suffer from two shortcomings: (a) they depend upon users setting up their own thresholds to be warned of phishing and (b) browser-based solutions are vulnerable to attacks that disable these defenses.

We propose an institution-wide, two-staged IDS system to detect phishing. Our solution performs an initial estimation of phishing email, followed by automated web crawlers visiting those potential phishing sites. Recursive crawling of potentially phishing sites increases the accuracy of the detection whether a web site impersonates another. Although our algorithms used in both stages are preliminary, our implemented framework can be used with any other algorithm in their place. Our initial results are promising with an overall detection time of 7.4 seconds. Our ongoing work is enhancing the detection algorithms to detect phishing sites. We are also working on using Darknets [19] to capture additional, potentially phishing emails.

References

1. The Anti-Phishing Working Group (APWG): http://www.antiphishing.org
2. Financial Service Technology Consortium (FSTC): North-America based financial institutions, technology vendors, independent research organizations and government agency, available at: http://www.fstc.org/projects/docs/FSTC_Counter_Phishing_Project_Whitepaper.pdf
3. Chen, Y., Ma, W.-Y., Zhang, H.-J.: Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices. In: WWW 2003 (May 20-24, 2003)
4. Fette, I., Sadeh, N., Thomasic, A.: Learning to Detect Phishing Emails. WWW (to appear, 2007), available at: http://www.cs.cmu.edu/ tomasic/doc/2007/FetteSadehTomasicWWW2007.pdf
5. Chou, N., Ledesma, R., Teraguchi, Y., Boneh, D., Mitchell, J.C.: Client-side defense against web-based identity theft (Webspoof), available at: http://www.crypto.stanford.edu/SpoofGuard/webspoof.pdf
6. Provos, N.: A Virtual Honeypot Framework. available at: http://www.niels.xtdnet.nl/papers/honeyd.pdf
7. Vanderavero, N., Brouckaert, X., Bonaventure, O., Charlier, B.L.: The HoneyTank.: A Scalable Approach to collect malicious Internet Traffic. In: international infrastructure survivability workshop (IISW'04) 2004, held in conjunction with the 25th IEEE International Real-time systems symposium (RTSS04), IEEE Computer Society Press, Los Alamitos (2004), available at: http://www.info.ucl.ac.be/people/OBO/papers/honeytank.pdf
8. Zhang, Y., Hong, J., Cranor, L.: CANTINA: A Content Based Approach to Detecting Phishing Sites. WWW (to appear, 2007), available at www.cups.cs.cmu.edu/trust.php
9. Wenyin, L., Huang, G., Xiaoyue, L., Min, Z., Deng, X.: Detection of Phishing Webpages based on Visual Similarity. WWW (May 10-14, 2005) Chiba, Japan (2005)
10. Cordero, A., Blain, T.: Catching Phish: Detecting Phishing Attacks From Rendered website Images. available at: http://www.cs.berkeley.edu/ asimma/294-fall06/ projects/ reports/cordero.pdf
11. RSS Feeds. available at: http://en.wikipedia.org/wiki/RSS_(file_format)
12. The Wdiff tool. available at: http://www.gnu.org/software/wdiff/wdiff.html
13. An HTML parser. available at: http://htmlparser.sourceforge.net/
14. The Apache SpamAssassin Project. available at: http://spamassassin.apache.org/
15. Apple Mac OS X Safari Browser. available at: http://www.apple.com/macosx/features/safari/
16. Safari Cocoa Plugin. Available at: http://developer.apple.com/internet/safari/
17. GNU Octave. Available at: http://www.gnu.org/software/octave/
18. ImageMagick. Available at: http://www.imagemagick.org/script/index.php
19. The Darknet Project. Available at: http://www.cymru.com/Darknet/
20. The R-Project. Available at: http://www.r-project.org/
21. Lucene Nutch. http://www.lucene.apache.org/nutch

A Three-Dimensional Conceptual Framework for Database Privacy

Josep Domingo-Ferrer

Rovira i Virgili University
UNESCO Chair in Data Privacy
Department of Computer Engineering and Mathematics
Av. Països Catalans 26, E-43007 Tarragona, Catalonia
josep.domingo@urv.cat

Abstract. Database privacy is an ambiguous concept, whose meaning is usually context-dependent. We give a conceptual framework for technologies in that field in terms of three dimensions, depending on whose privacy is considered: i) respondent privacy (to avoid re-identification of patients or other individuals to whom the database records refer); ii) owner privacy (to ensure that the owner must not give away his dataset); and iii) user privacy (to preserve the privacy of queries submitted by a data user). Examples are given to clarify why these are three *independent* dimensions. Some of the pitfalls related to combining the privacy interests of respondents, owners and users are discussed. An assessment of database privacy technologies against the three dimensions is also included.

Keywords: Statistical database privacy, Private information retrieval, Privacy-preserving data mining, Security and privacy of electronic health records.

1 Introduction

The meaning of database privacy is largely dependent on the context where this concept is being used. In official statistics, it normally refers to the privacy of the respondents to which the database records correspond. In co-operative market analysis, it is understood as keeping private the databases owned by the various collaborating corporations. In healthcare, both of the above requirements may be implicit: patients must keep their privacy and the medical records should not be transferred from a hospital to, say, an insurance company. In the context of interactively queryable databases and, in particular, Internet search engines, the most rapidly growing concern is the privacy of the queries submitted by users (especially after scandals like the August 2006 disclosure by the AOL search engine of 36 million queries made by 657000 users). Thus, what makes the difference is whose privacy is being sought.

The last remark motivates splitting database privacy in the following three dimensions:

1. *Respondent privacy* is about preventing re-identification of the respondents (*e.g.* individuals like patients or organizations like enterprises) to which the records of a database correspond. Usually respondent privacy becomes an issue only when the database is to be made available by the data collector (hospital or national statistical office) to third parties, like researchers or the public at large.
2. *Owner privacy* is about two or more autonomous entities being able to compute queries across their databases in such a way that only the results of the query are revealed.
3. *User privacy* is about guaranteeing the privacy of queries to interactive databases, in order to prevent user profiling and re-identification.

The technologies to deal with the above three privacy dimensions have evolved in a fairly independent way within research communities with surprisingly little interaction:

– Respondent privacy is pursued mainly by statisticians and a few computer scientists working in statistical disclosure control (SDC), also known as statistical disclosure limitation (SDL) or inference control [17,26].
– Owner privacy is the primary though not the only goal[1] of privacy-preserving data mining (PPDM, [5]), a discipline born in the database and data mining community. Interestingly enough, the term privacy-preserving data mining independently and simultaneously appeared in the cryptographic community [18,19] to denote a special case of secure multiparty computation where each party holds a subset of the records in a database (horizontal partitioning).
– Finally, user privacy has found solutions mainly in the cryptographic community, where the notion of private information retrieval was invented (PIR, [8]).

Only quite recently some researchers have started to look for holistic privacy solutions, but to the best of our knowledge no comprehensive technology covering the three dimensions above exists yet. In [6] the apparent conflict between respondent privacy and user privacy is highlighted: it seems necessary for the data owner to analyze the user queries in order to check that they will not result in disclosure of sensitive respondent data. In [3], a new technology called hippocratic databases is described which seeks to ensure respondent privacy and owner privacy; examples related to healthcare are provided, with respondents being patients and data owners being hospitals.

1.1 Contribution and Plan of this Paper

The general aim of this paper is to clarify the independent nature of the privacy of respondents, owners and users in databases. Specifically, it will be shown

[1] In [13], a PPDM approach based on randomized responses is presented whose primary aim is claimed to be respondent privacy; however, typical respondents are unlikely to have or use the proposed randomizing device when answering a survey, whereas data owners could make use of it to protect their own privacy.

that guaranteeing privacy for one of those entities does not ensure privacy for the other two. These three privacy dimensions constitute a conceptual framework that can be used to classify technologies based on whose privacy they offer.

In Section 2 we show that respondent privacy and owner privacy are independent dimensions. Section 3 illustrates the independence of respondent privacy and user privacy. Section 4 deals with the independence of owner privacy and user privacy. A tentative assessment of technologies according to the three privacy dimensions is given in Section 5. Section 6 contains some conclusions on the simultaneous satisfaction of the three privacy dimensions as well as topics for future research.

2 Independence of Respondent Privacy vs Owner Privacy

If a dataset is published without any anonymization masking, in general it will violate both respondent and owner privacy. However, there are more interesting cases.

Respondent privacy without owner privacy. Consider the patient toy dataset 1 in Table 1 (left). Assume that the records have been obtained by a pharmaceutical company which is testing a new drug against hypertension. All patients in the dataset suffered from hypertension before starting the treatment. Direct identifiers have been suppressed, but height and weight constitute key attributes in the sense of [9,20], *i.e.* they identify the respondent with some degree of ambiguity [2]: an intruder can easily gauge the height and weight of an individual he knows in order to link the identity of that individual to a record in the dataset. The remaining attributes (systolic blood pressure and AIDS) are confidential attributes.

Luckily enough for the patients in Dataset 1 of Table 1, the dataset turns out to spontaneously satisfy k-anonymity [21,20,23] for $k = 3$ with respect to the key attributes (height, weight). In other words, each combination of the key attributes appears at least 3 times. Thus, if 3-anonymity is considered to be enough protection for patients, Dataset 1 offers respondent privacy and could be published as far as the patients are concerned [3].

However, since Dataset 1 is the one actually obtained in the clinical drug trial, the pharmaceutical company is unwilling to share those data with possible

[2] Height and weight make sense as key attributes only in small populations. In large populations, there are many individuals sharing similar weights and heights. However, we keep those two attributes as example key attributes for the sake of clarity.

[3] If records sharing a combination of key attributes in a k-anonymous dataset also share the values for one or more confidential attributes, then k-anonymity does not guarantee respondent privacy. A stronger property called p-sensitive k-anonymity [24] is in general required: there should be at least p distinct values of each confidential attribute within each group of records sharing a combination of key attributes.

Table 1. Left, patient data set no. 1. Right, patient data set no. 2.

Height (cm)	Weight (kg)	Blood pressure (syst, mmHg)	AIDS (Y/N)	Height (cm)	Weight (kg)	Blood pressure (syst, mmHg)	AIDS (Y/N)
175	76	117	Y	160	110	146	N
175	76	131	N	170	65	117	Y
175	76	122	N	173	75	131	N
180	81	115	N	175	80	122	N
180	81	122	Y	180	68	115	N
180	81	146	N	183	81	122	Y
190	95	110	N	187	95	110	N
190	95	115	Y	190	95	115	Y
190	95	125	N	192	99	125	N
190	95	140	N	192	101	140	N

competitors. Therefore publication of the dataset is compatible with respondent privacy but violates the owner privacy.

Respondent privacy and owner privacy. If a dataset is adequately masked before release, then both owner and respondent privacy are obtained without significantly damaging the utility of the data for designated user analyses. There are plenty of examples in the literature along this line, some of which are:

– In [5], noise addition is used to mask an original dataset for owner privacy and, to a large extent, for respondent privacy. Regarding utility, the distribution of the original dataset can still be reconstructed from the noise-added data, so that decision-tree classifiers properly run on the masked data.
– In [1], masking through condensation (actually a special case of multivariate microaggregation, [10]) is proposed to achieve privacy-preserving data mining. Since the covariance structure of the original attributes is preserved, a variety of analyses can be validly carried out by users on the masked data. Since microaggregation/condensation with minimum group size k on the key attributes guarantees k-anonymity ([12]), the approach in [1] can also guarantee respondent privacy.

Hippocratic databases [4,3] mentioned above are a real-world technology integrating k-anonymization for respondent privacy and PPDM based on noise addition [15] for owner privacy.

Owner privacy without respondent privacy. Imagine that the patient dataset obtained by the pharmaceutical company is not Dataset 1 in Table 1, but Dataset 2. The new dataset is no longer 3-anonymous with respect to the key attributes (height, weight). Therefore, releasing a single record is a violation of respondent privacy: the patient's blood pressure and AIDS condition could be linked to his/her identity by means of the unique patient's key attributes. In fact, merely revealing the name of a patient who took part in the trial already discloses that he/she suffers from hypertension (only patients with hypertension underwent the trial).

However, neither revealing a single record nor the name of someone who took part in the trial can be said to violate the data owner's privacy (especially if the dataset is large). Thus, we can have owner privacy without respondent privacy.

A subtler example is conceivable with the method proposed in [5] and mentioned above. In [11] it is shown that, for higher-dimensional datasets, the property of the method in [5] that the distribution of the original data can be reconstructed from the noise-added data can result in violation of respondent privacy. The reason is that, for higher dimensions, data tend to become sparse, *i.e.* with a lot of rare combinations of attribute values: if the reconstructed distribution fits the multidimensional histogram of the original data too well, rare combinations in the original data are disclosed. This is a non-trivial case of owner privacy without respondent privacy.

3 Independence of Respondent Privacy and User Privacy

The trivial case with neither respondent nor user privacy is the most common one: a queryable database where neither records nor user queries undergo any anonymization (in particular, this is the case of Internet search engines). Situations with at least respondent or user privacy are discussed in the next subsections.

Respondent privacy without user privacy. The conflict between respondent privacy and user privacy is apparent in statistical disclosure control of interactively queryable statistical databases. The scenario is a database to which the user can submit statistical queries (sums, averages, etc.). The aggregate information obtained by a user as a result of successive queries should not allow him to infer the values of confidential attributes for specific individuals (respondent privacy). Currently employed strategies rely on perturbing, restricting or replacing by intervals the answers to certain queries. Examples of those three strategies can be found in [7,14,16], respectively.

All SDC methods for interactive statistical databases assume that the data owner operating the database exactly knows the queries submitted by users. This knowledge is deemed necessary to check that users do not submit a series of queries designed to isolate a single record in the database. Thus, there is no user privacy whatsoever. Even without user privacy, the SDC problem in this kind of databases is known to be difficult since the 1980s, due to the existence of the tracker attack [22].

Respondent privacy and user privacy. If the records in an interactively queryable statistical database are k-anonymous (spontaneously as in Dataset 1 or after a k-anonymization process as described in [2,12]), then no user query can jeopardize respondent privacy. In this case, the use of private information retrieval protocols to preserve the privacy of user queries can be afforded.

User privacy without respondent privacy. This situation is the most likely one if private information retrieval is allowed on unmasked records. To illustrate, assume that PIR is offered on Dataset 2 in Table 1. Even if allowed queries are only of statistical nature, a user could take advantage of PIR to submit the following queries (assuming PIR protocols existed for those query types):

```
SELECT COUNT(*) FROM Dataset 2 WHERE height < 165 AND weight > 105
SELECT AVG(blood_pressure) FROM Dataset 2 WHERE height < 165 AND
weight > 105
```

The first query tells the user that there is only one individual in the dataset smaller than 165 cm and heavier than 105 kg. With this knowledge, the user can establish that the average blood pressure 146 returned by the second query corresponds to that single individual, who turns out to be someone suffering from serious hypertension. Re-identifying such a small and heavy individual as Mr./Mrs. X should not be too difficult. If the user is an insurance company, Mr./Mrs. X might see his/her life insurance application rejected or accepted only at an extremely high premium.

4 Independence of Owner Privacy and User Privacy

If a database owner allows unrestricted queries on original data and user queries are not protected, there is neither owner privacy nor user privacy. The cases with at least one of both properties are next discussed.

Owner privacy without user privacy. PPDM methods developed in the cryptographic community in the spirit of the seminal paper [18] are special cases of secure multiparty computation. The idea is that two or more parties owning confidential databases run a data mining algorithm (*e.g.* a classifier) on the union of their databases, without revealing any unnecessary information.

Thus, the users coincide with the data owners. However, the focus is on ensuring the privacy of the data owned by each party (owner privacy). The analysis or data mining algorithm run by the parties is known to all of them. Indeed, as usual in secure multiparty computation, all parties interactively co-operate to obtain the result of the analysis. This is hardly compatible with private information retrieval for user privacy. Thus, we can conclude that cryptographic PPDM offers owner privacy but no user privacy.

Non-cryptographic PPDM methods developed in the data mining community are friendlier toward user privacy, as will be discussed in the next paragraph below. However, application of those methods without PIR also leads to owner privacy without user privacy.

Owner privacy and user privacy. Unlike cryptographic PPDM, non-cryptographic PPDM developed by data miners is usually non-interactive. The data owner first protects his data and then accepts queries on them. Whatever the case, the data owner does not need to know the exact query being computed on his protected data, so that PIR for user privacy is compatible with non-cryptographic PPDM.

One must acknowledge here that, while some PPDM methods like [2] allow a broad range of analyses/queries to be performed on the protected data, other methods have been designed to support a specific class of analyses on the privacy-protected data (*e.g.* [5] is designed for decision-tree classifiers and methods in [25] are designed for association rule mining). However, even with the latter methods, the data owner does not need to know anything about the exact user analyses beyond the (likely) fact that they belong to the class supported by the PPDM algorithm.

User privacy without owner privacy. This is the situation if unrestricted PIR queries are allowed by an owner on his original data. If the database is a public one and contains non-confidential information, this is the most desirable situation. For example, in the context of Internet search engines, user privacy is arguably the only privacy that should be cared about.

5 Tentative Technology Scoring

In order to demonstrate the usefulness of the proposed three-dimensional conceptual framework, we attempt as an exercise a scoring of the *non-exhaustive* list of privacy technologies mentioned in this paper. Table 2 is an assessment of how well each technology class performs in each privacy dimension. This scoring is qualitative and tentative, in that we base our assessment on the usual claims of each technology class, rather than on the actual properties of specific proposals within each class. For the reader's orientation, we mean by SDC the methods in [17,26]; example proposals of use-specific non-crypto PPDM are [5,25]; an example generic non-crypto PPDM method is [2]; an example PIR method is [8].

Table 2. Tentative scoring of technology classes

Technology class	Respondent privacy	Owner privacy	User privacy
SDC	medium-high	medium	none
Use-specific non-crypto PPDM	medium	medium-high	none
Generic non-crypto PPDM	medium	medium-high	none
Crypto PPDM	high	high	none
PIR	none	none	high
SDC + PIR	medium-high	medium	high
Use-specific non-crypto PPDM + PIR	medium	medium-high	medium
Generic non-crypto PPDM + PIR	medium	medium-high	high

The rationale for the grades in Table 2 follows:

– Being based on multi-party secure computation, crypto PPDM methods are the PPDM methods offering highest owner privacy. As a side property, they also offer respondent privacy (records in the database are not leaked). In comparison, non-crypto PPDM only offers medium-high owner privacy; however, as argued above, it is more flexible and it can be combined with PIR.

- A distinction is made between use-specific and generic non-crypto PPDM: when use-specific non-crypto PPDM is combined with PIR, there is some clue on the queries made by the user (they are likely to correspond to the uses the PPDM method is intended for); therefore generic non-crypto PPDM is better for combination with PIR in view of attaining high user privacy.
- Non-crypto PPDM and SDC in Table 2 are assumed to rely on data masking, rather than on query control.
- If non-crypto PPDM perturbs the data, it normally provides some level of respondent privacy in addition to owner privacy.
- Similarly, SDC masking normally provides some level of owner privacy in addition to respondent privacy.

Note that the last two assertions do not contradict the independence between respondent and owner privacy, justified in Section 2 above.

6 Conclusions and Future Research

Respondent privacy, owner privacy and user privacy have been shown to be independent, yet compatible properties. Even though satisfying one of them gives no assurance about the others, we can state a few lessons learned which can be used as guidelines to simultaneous fulfillment of the three privacy dimensions:

- Respondent privacy relies on data masking (*e.g.* for k-anonymity) or on query control (needed if interactive queries against original databases are allowed). Since query control is hardly compatible with user privacy, data masking must be used for respondent privacy if the latter property must live together with user privacy.
- Owner privacy relies on cryptographic or non-cryptographic PPDM. Being based on interactive multiparty computation, cryptographic PPDM assumes that the joint computation being carried out is known to all parties, which is not compatible with user privacy. Therefore, non-cryptographic PPDM seems a wiser choice if owner privacy is to be made compatible with user privacy.
- Most forms of non-cryptographic PPDM rely on perturbing the original data. If this perturbation is such that the underlying data are k-anonymized (as in [2,12]), then owner and respondent privacy are simultaneously achieved.

Hence, one possible way to fulfill the three privacy dimensions is for a database which is not originally k-anonymous to be k-anonymized (via microaggregation-condensation, recoding, suppression, etc.) and to be added a PIR protocol to protect user queries.

Future research should explore other possible solutions satisfying the privacy of respondents, owners and users. Also, the impact on data utility of offering the three dimensions of privacy (rather than just one or two of them) should be investigated. An interesting challenge is to offer privacy for everyone without incurring extra data utility penalties.

Disclaimer and Acknowledgments

The author is solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization. This work was partly supported by the Spanish Ministry of Education through projects SEG2004-04352-C04-01 "PROPRIETAS" and CONSOLIDER CSD2007-00004 "ARES", and by the Government of Catalonia under grant 2005 SGR 00446. Thanks go to Agusti Solanas for his comments on a draft version of this paper.

References

1. Aggarwal, C.C., Yu, P.S.: A condensation approach to privacy preserving data mining. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., Ferrari, E. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 183–199. Springer, Heidelberg (2004)
2. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: k-Anonymity: Algorithms and hardness. Technical report, Stanford University (2004)
3. Agrawal, R., Grandison, T., Johnson, C., Kiernan, J.: Enabling the 21st century health care information technology revolution. Communications of the ACM 50(2), 35–42 (2007)
4. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Hippocratic databases. In: Proceedings of the 28th International Conference on Very Large Databases, Hong Kong (2002)
5. Agrawal, R., Srikant, R.: Privacy preserving data mining. In: Proceedings of the ACM SIGMOD, pp. 439–450. ACM Press, New York (2000)
6. Aguilar, C., Deswarte, Y.: Single database private information retrieval schemes. In: Domingo-Ferrer, J., Franconi, L. (eds.) PSD 2006. LNCS, vol. 4302, pp. 257–265. Springer, Heidelberg (2006)
7. Chin, F.Y., Ozsoyoglu, G.: Auditing and inference control in statistical databases. IEEE Transactions on Software Engineering E-8, 574–582 (1982)
8. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: IEEE Symposium on Foundations of Computer Science (FOCS), pp. 41–50. IEEE Computer Society Press, Los Alamitos (1995)
9. Dalenius, T.: Finding a needle in a haystack - or identifying anonymous census records. Journal of Official Statistics 2(3), 329–336 (1986)
10. Domingo-Ferrer, J., Mateo-Sanz, J.M.: Practical data-oriented microaggregation for statistical disclosure control. IEEE Transactions on Knowledge and Data Engineering 14(1), 189–201 (2002)
11. Domingo-Ferrer, J., Sebé, F., Castellà, J.: On the security of noise addition for privacy in statistical databases. In: Domingo-Ferrer, J., Torra, V. (eds.) PSD 2004. LNCS, vol. 3050, pp. 149–161. Springer, Heidelberg (2004)
12. Domingo-Ferrer, J., Torra, V.: Ordinal, continuous and heterogeneous k-anonymity through microaggregation. Data Mining and Knowledge Discovery 11(2), 195–212 (2005)
13. Du, W., Zhan, Z.: Using randomized response techniques for privacy-preserving data mining. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, D.C, pp. 505–510 (2003)

14. Duncan, G.T., Mukherjee, S.: Optimal disclosure limitation strategy in statistical databases: deterring tracker attacks through additive noise. Journal of the American Statistical Association 95, 720–729 (2000)
15. Evfimievski, A.: Randomization in privacy-preserving data mining. SIGKDD Explorations: Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining 4(2), 43–48 (2002)
16. Gopal, R., Garfinkel, R., Goes, P.: Confidentiality via camouflage: the cvc approach to disclosure limitation when answering queries to databases. Operations Research 50, 501–516 (2002)
17. Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Lenz, R., Longhurst, J., Schulte-Nordholt, E., Seri, G., DeWolf, P.-P.: Handbook on Statistical Disclosure Control (version 1.0). In: Eurostat (CENEX SDC Project Deliverable) (2006)
18. Lindell, Y., Pinkas, B.: Privacy preserving data mining. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 36–53. Springer, Heidelberg (2000)
19. Lindell, Y., Pinkas, B.: Privacy preserving data mining. Journal of Cryptology 15(3), 177–206 (2002)
20. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
21. Samarati, P., Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International (1998)
22. Schlörer, J.: Disclosure from statistical databases: quantitative aspects of trackers. ACM Transactions on Database Systems 5, 467–492 (1980)
23. Sweeney, L.: k-Anonimity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge Based Systems 10(5), 557–570 (2002)
24. Truta, T.M., Vinay, B.: Privacy protection: p-sensitive k-anonymity property. In: 2nd International Workshop on Privacy Data Management PDM 2006, p. 94. IEEE Computer Society Press, Los Alamitos (2006)
25. Verykios, V.S., Elmagarmid, A.K., Bertino, E., Saygin, Y., Dasseni, E.: Association rule hiding. IEEE Transactions on Knowledge and Data Engineering 16(4), 434–447 (2004)
26. Willenborg, L., DeWaal, T.: Elements of Statistical Disclosure Control. Springer, New York (2001)

Novel RFID Authentication Schemes for Security Enhancement and System Efficiency

N.W. Lo and Kuo-Hui Yeh

Department of Information Management
National Taiwan University of Science and Technology
43 Sect. 4 Keelung Rd., Taipei, 106 Taiwan, R.O.C.
Fax number: 886-2-2737-6777
nwlo@cs.ntust.edu.tw
D9409101@mail.ntust.edu.tw

Abstract. As industries aggressively deploy Radio Frequency IDentification application systems, the user privacy invasion and system security threats are increasingly concernd by individuals and organizations. Recently several hash-based mutual authentication schemes have been proposed to resolve security-related problems. However, previous schemes either suffer from security loopholes or have system efficiency problem for identity match process. In this paper, the security flaws of two recently proposed hash-based authentication schemes are analyzed at first. Based on this analysis, we identify the security and privacy criterions for the authentication process of RFID systems, and propose a new mutual authentication scheme to eliminate possible security flaws and enhance privacy protection to the owner of an object with RFID tag attached on it. In addition, we develop an efficient identity match and retrieval mechanism to relieve the heavy computation load of traditional match scheme between received tag identity and records in backend database of RFID systems.

Keywords: RFID; Authentication; Privacy; Security.

1 Introduction

As Radio Frequency IDentification (RFID) technology provides an efficient and accurate way to identify physical resources and at the same time preserves very attractive deployment characteristics to industries such as simple system installation and deployment process, wireless accessibility and low-cost manufacture, RFID has been considered as the next generation technology for object identification and management in a ubiquitous network environment. In order to robustly perform identity validation and access control in RFID systems, an efficient and reliable authentication scheme is definitely required.

The existing RFID systems are vulnerable to many security attacks and potential privacy threats due to the nature of restricted computation ability and limited memory space of low-cost RFID tags. For example, RFID-tagged goods in a chain store make themselves easily been scanned by industrial espionage activities to gather unprotected goods information and even trace customer preferences. More sophisticated security threats such as constellation attack, breadcrumb threat and RFID tag cloning are also raised in the recent years. To achieve robust security and privacy requirements in RFID systems, an authentication scheme using hash function or cryptographic unit such as AES encryption at both the tag side and the reader/server side becomes a promising candidate. Ohkubo et al. in [2] pointed out that in a 5-cent passive tag the number of gates available for security design cannot exceed 2.5 to 5 K gates. In addition, according to K. Yuksel in [4], a hash function unit with the block size of 64-bits can be implemented under 1.7 K gates. From these observations, we assume that a one-way hash function circuit unit can be implemented in a RFID tag with reasonable cost.

Recently the study of hash-based RFID authentication schemes [1-3, 5-9] has developed rapidly and the accumulated achievements have drawn scholars' attention. However, most of previous schemes suffer from different kinds of security attacks, privacy-related problems, and computation-consuming identity match process at the backend server. Hence, the main purpose of this study is to remedy the security flaws of previous schemes by introducing a novel mutual authentication scheme with timestamp checking mechanism and one-way hash function, and provide an efficient identity match mechanism for the backend server.

2 Related Work

Recently, much attention has been directed to hash-based RFID authentication research due to the potential capability on security enhancement. Weis et al. first proposed two authentication mechanisms in [1], called hash-based access control and randomized hash-locking access control, to achieve security and privacy aspects for RFID systems. However, since the wireless communication channel is insecure, the attacker can manipulate the communication process during authentication to easily break their proposed scheme with historical information gained from eavesdropping. Because the brute-force identity search mechanism is adopted for every query in their scheme, the backend server will encounter heavier computation load. Ohkubo et al. in [2] proposed an authentication scheme with hashing chain mechanism. Two important security requirements, indistinguishability and forward security, were claimed to be achieved by their scheme. Unfortunately, this authentication mechanism cannot prevent the replay attack. Henrici and Műller in [3] used varying identifications at each authentication session to enhance location privacy in their RFID authentication scheme. However, their scheme does not support the anonymity property because the tag always responds a reader's query with the same hashed value of its secret identification until it updates its secret identification at the end of this successful authentication session. Yang et al. in [6] remedied the security weakness of Henrici and Műller's scheme [3] by introducing a new authentication scheme to resolve the location tracking problem in RFID systems. Unfortunately, their scheme cannot provide privacy protection to the tag carrier.

In [7], An and Oh proposed a user privacy-aware authentication process which utilizes hash function and random number generator. Due to the lack of replay attack prevention mechanism, their scheme can be broken by sending eavesdropped historical messages. In their scheme the backend server also spends more computation power on the backend identity match process. Kim et al. in [9] proposed a RFID authentication scheme to enhance location privacy and support forward security via dynamically updating the secret information with specific generated stream blocks. However, the anonymity property for RFID tags is not provided in their scheme because the tag identification can be derived by eavesdropping multiple messages during various authentication communications. In addition, this scheme cannot defend against replay attack and also adopt the traditional identity match mechanism at the backend server.

In November 2006, two hash-based authentication schemes for security improvement in RFID systems are proposed by Osaka et al. in [8] and Park & Lee in [5]. Osaka et al. developed an authentication scheme with ownership transfer concept to support the ownership privacy when the ownership of RFID-tagged items is changed. In the meantime, Park & Lee developed a novel authentication scheme to provide stronger security for RFID systems. However, both of schemes still have security weaknesses which will be analyzed in the following.

In terms of security analysis on Osaka et al's authentication scheme [8], we report our observation as follows. First, if one legitimate but malicious reader always sends the same random number r with its query command to a specific tag, the tag will always response the same value a until it receives update message e from the backend server. This repeatable behavior pattern enables adversaries to trace the tag. In addition, the replay attack cannot be prevented by the scheme because the backend server does not support mechanisms to tell whether an incoming message is a replay or not. In their scheme the tag does not have the verification mechanism against incoming update message e; an attacker can send a fake update message e' without being noticed which will result in the shared secret identities between the tag and the backend server out of synchronization (i.e., *DoS* attack). Regarding to forward security, we show that Osaka et al's scheme cannot protect the historical journey trajectory of a RFID tag in case the tag is compromised during the $i+1$-th authentication session. Based on this assumption, the attacker already gets the secret identity $E_{k\_(i+1)}(ID)$. Consider that the attacker has eavesdropped and stored all previous transmitted messages a_i, e_i and r_i of each session i. The attacker can iteratively apply the following computations to easily retrieve all previously sent secret identities and messages: $E_{k\_i}(ID)=E_{k\_(i+1)}(ID)\oplus e_i$ and $a_i= H(E_{k\_i}(ID)\oplus r_i)$. Hence, the forward security requirement is not fulfilled. Finally, their secret identity match mechanism at the backend server can consume a lot of computation resources and execution time if the total number of tag entries in the database is quite large.

Park and Lee's scheme [5] could be better convinced after resolving a couple of security flaws. First of all, a tag with the same $H(SID_i)$ value will be easily traced by a legitimate but malicious reader through a normal authentication behavior. Secondly, the backend server cannot detect a replay attack. In Figure 1, we show another security flaw of their scheme. If a legitimate but malicious reader sequentially invokes two queries to one specific tag in a reasonable period of time before the tag updates its security identification (SID_i), then the tag will response two authentication requests A_1 and A_2 correspondingly. The attacker can XOR the two *count_state* values in A_1 and A_2 to eliminate the *T_key*; based on the known value R, the security identification can be derived by the malicious reader. In that case, the anonymity property is violated. In addition, once the tag was compromised by an attacker (i.e., the *T_key* and *R_Key* will be known), the attacker can derive the random number R of each authentication session i by XOR the eavesdropped R_Value and R_Key from each authentication session i. With the derived security identification SID_i of each authentication session i, all past secret information, such as $T\_value=SID_i \oplus T\_key$, $count=count\_state \oplus T\_value$ and $C=H(flag||T\_value||count||R)$, can be computed by the attacker. Therefore, forward security cannot be guaranteed in their scheme either. Finally, in their scheme malicious attacks or malfunction condition such as message transmission interruption, will switch the normal authentication process ($flag$=0) into exception situation ($flag$=1). The authentication process under exception situation ($flag$=1) will result in heavy computation load on security identification matching process by computing the value $T\_ID'=T\_key \oplus R$ and comparing $H(T\_ID)'=H(T\_ID)$ for every tuple at the backend database.

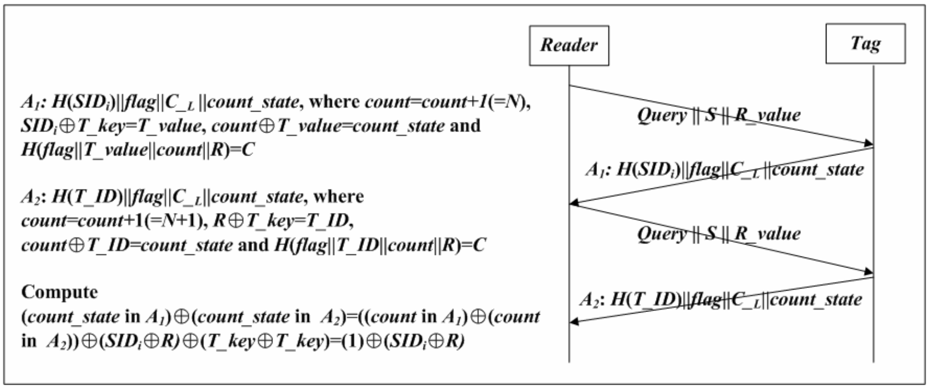

Reader
Tag
A_1: $H(SID_i)||flag||C\_L||count\_state$, where $count=count+1(=N)$, $SID_i \oplus T\_key=T\_value$, $count \oplus T\_value=count\_state$ and $H(flag||T\_value||count||R)=C$
A_2: $H(T\_ID)||flag||C\_L||count\_state$, where $count=count+1(=N+1)$, $R \oplus T\_key=T\_ID$, $count \oplus T\_ID=count\_state$ and $H(flag||T\_ID||count||R)=C$
Compute ($count\_state$ in A_1)$\oplus$($count\_state$ in A_2)=(($count$ in A_1)$\oplus$($count$ in A_2))$\oplus(SID_i \oplus R) \oplus (T\_key \oplus T\_key)=(1) \oplus (SID_i \oplus R)$
Query || S || R_value
A_1: $H(SID_i)||flag||C\_L||count\_state$
Query || S || R_value
A_2: $H(T\_ID)||flag||C\_L||count\_state$

Fig. 1. The illustration of security weakness in Park & Lee's scheme

3 Proposed Solution

In this section, based on hash function and simple XOR operation, we develop two novel schemes for RFID systems. In Section 3.1 we depict a new authentication scheme to remedy the security weaknesses of previous researches. The second one, called *Efficient Identity Match* scheme, is illustrated in Section 3.2 to pursue better system performance on data matching at the backend database.

3.1 New Authentication Scheme

In our protocol, we assume the tag is vulnerable to be compromised. The secret information inside a tag, such as shared secret key and security identification, can be retrieved and modified by the attacker once that tag is compromised. In contrast with previous researches, insecure channels are assumed through the whole communication environment of a RFID system.

Each tag, denoted as Tag_x, is given two random secrets, the shared key value T_key and the security identification during the i-th session SID_i, which are randomly generated and shared with the backend server. Tag_x also maintains two extra values, $flag$ and reader's shared key R_key. The backend server maintains a record for each tag including associated reader's password *R_Key*, the shared security identification SID_{i_DB}, the shared secret key $T\_key_{\_DB}$ and the timestamp of the last incoming legitimate message T_{i_DB}. The security identification SID_i will be updated at both the backend server and the corresponding tag after each successful authentication. For the sake of simplicity, the initial values of *flag* and T_{i_DB} are set as zero here. The security identifications associated with Tag_x and backend database are both initially set as SID_{x_0}.

The proposed scheme considers two different situations based on previous authentication session is safely terminated (*flag*=0) or not (*flag*=1). We depict the detailed procedure in Figure 2.

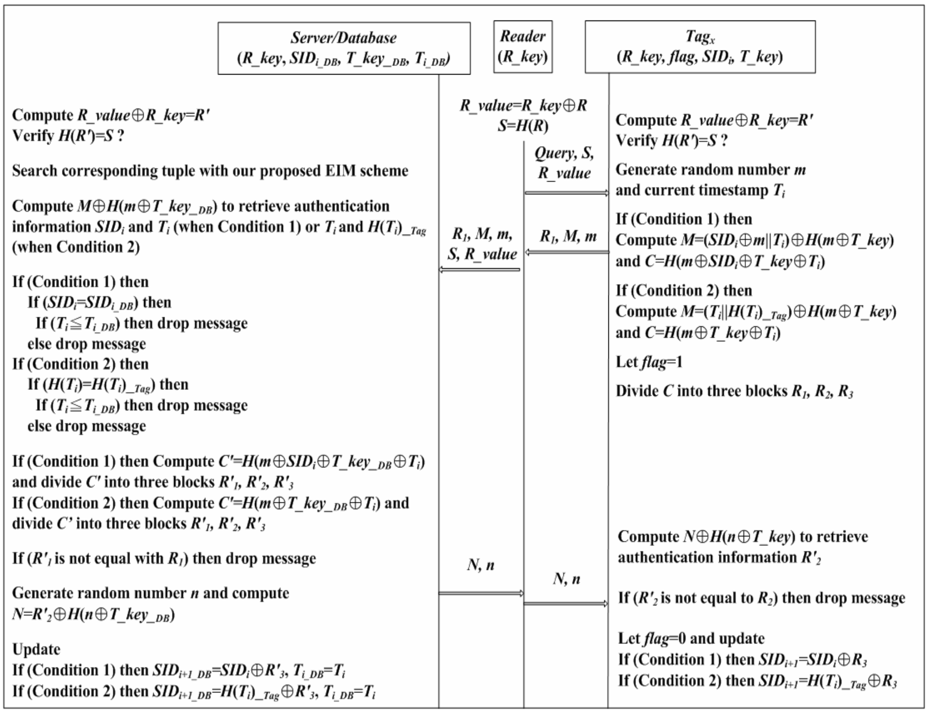

Server/Database
(R_key, SID_i_DB, T_key_DB, T_i_DB)
Reader
(R_key)
Tag_x
(R_key, flag, SID_i, T_key)
Compute R_value⊕R_key=R'
Verify H(R')=S ?
Search corresponding tuple with our proposed EIM scheme
Compute M⊕H(m⊕T_key_DB) to retrieve authentication information SID_i and T_i (when Condition 1) or T_i and H(T_i)_Tag (when Condition 2)
If (Condition 1) then
If (SID_i=SID_i_DB) then
If (T_i≦T_i_DB) then drop message
else drop message
If (Condition 2) then
If (H(T_i)=H(T_i)_Tag) then
If (T_i≦T_i_DB) then drop message
else drop message
If (Condition 1) then Compute C'=H(m⊕SID_i⊕T_key_DB⊕T_i) and divide C' into three blocks R'_1, R'_2, R'_3
If (Condition 2) then Compute C'=H(m⊕T_key_DB⊕T_i) and divide C' into three blocks R'_1, R'_2, R'_3
If (R'_1 is not equal with R_1) then drop message
Generate random number n and compute N=R'_2⊕H(n⊕T_key_DB)
Update
If (Condition 1) then SID_i+1_DB=SID_i⊕R'_3, T_i_DB=T_i
If (Condition 2) then SID_i+1_DB=H(T_i)_Tag⊕R'_3, T_i_DB=T_i
R_value=R_key⊕R
S=H(R)
Query, S, R_value
R_1, M, m, S, R_value
R_1, M, m
N, n
N, n
Compute R_value⊕R_key=R'
Verify H(R')=S ?
Generate random number m and current timestamp T_i
If (Condition 1) then
Compute M=(SID_i⊕m||T_i)⊕H(m⊕T_key) and C=H(m⊕SID_i⊕T_key⊕T_i)
If (Condition 2) then
Compute M=(T_i||H(T_i)_Tag)⊕H(m⊕T_key) and C=H(m⊕T_key⊕T_i)
Let flag=1
Divide C into three blocks R_1, R_2, R_3
Compute N⊕H(n⊕T_key) to retrieve authentication information R'_2
If (R'_2 is not equal to R_2) then drop message
Let flag=0 and update
If (Condition 1) then SID_i+1=SID_i⊕R_3
If (Condition 2) then SID_i+1=H(T_i)_Tag⊕R_3

Fig. 2. The proposed authentication scheme

Condition 1: previous authentication session is safely terminated (*flag=0*)

1. *Reader* → *Tag*$_x$: *Query, S, R_value*

Reader generates a random number *R*, performs XOR operation on it with the shared secret key $R\_value= R\_key \oplus R$ and applies the hash function on it $S=H(R)$. *Reader* then sends (*S, R_value*) with a *Query* signal as a challenge to Tag_x.

2. *Tag*$_x$ → *Reader*: R_1, *M, m*

Tag_x first performs an XOR operation onto *R_value* and *R_key* to retrieve the value R' and verifies whether $H(R')$ is equal to *S*. Once the authentication of the legitimate reader is passed, Tag_x sets the value *flag* to 1. Next, Tag_x generates a random number *m* and the current timestamp T_i to compute $M=((SID_i \oplus m)||T_i) \oplus H(m \oplus T\_key)$ and

$C=H(m\oplus SID_i\oplus T\_key\oplus T_i)$, where T_i is used to detect the replay attack. After dividing C into three blocks R_1, R_2 and R_3, Tag_x sends (R_1, M, m) as a response to *Reader*.

3. ***Reader*** **→** ***Server*****:** ***R_1, M, m, S, R_value***

Reader transmits an authentication request (R_1, M, m, S, R_value) to *Server*.

4. ***Server*** **→** ***Reader*****:** ***N, n***

When *Server* receives the authentication request, it firstly verifies whether $H(R')$ and S are identical as mentioned in step 2. According to our proposed EIM scheme in Section 3.2, *Server* can efficiently find the corresponding data tuple associated with Tag_x from the database. Next, *Server* computes $M\oplus H(m\oplus T\_key\__{DB})$ to retrieve security identification SID_i and the timestamp T_i. In order to verify the incoming message, *Server* applies the following two verification processes: (1) whether the values SID_i and SID_{i_DB} are the same, and (2) whether the timestamp is valid, $T_i>T_{i\_DB}$ (i.e., replay attack examination). If the incoming message is legitimate, *Server* calculates $C'=H(m\oplus SID_i\oplus T\_key\__{DB}\oplus T_i)$ and divides C' into three blocks R'_1, R'_2 and R'_3 to verify whether R'_1 is equivalent to R_1. If R'_1 and R_1 are identical, all contents of the incoming message are denoted as correct. *Server* computes $N=R'_2\oplus H(n\oplus T\_key\__{DB})$ with a generated random number n. Finally, *Server* sends (N, n) to *Reader*. At the same time, *Server* updates the shared secret information $SID_{i+1\_DB}=SID_i\oplus R'_3$ and $T_{i\_DB}=T_i$. Note that *Server* can send the information of tag or tagged item to *Reader* through a symmetric or asymmetric encryption.

5. ***Reader*** **→** ***Tag_x*****:** ***N, n***

Reader forwards (N, n) to Tag_x. For the validity of the incoming message, Tag_x computes $N\oplus H(n\oplus T\_key)$ to retrieve R'_2 and verifies whether R'_2 is equal to R_2. If Tag_x successfully completes the verification process, Tag_x will change the *flag* value from 1 to 0 and update the value of security identification SID_{i+1} to $SID_i\oplus R_3$.

Condition 2: previous authentication session is not safely terminated (*flag=1*)

1. ***Reader*** **→** ***Tag_x*****:** ***Query, S, R_value***

Same as the step 1 where *flag=0*.

2. ***Tag_x*** **→** ***Reader*****:** ***R_1, M, m***

The verification process of *Reader* is the same as step 2 where *flag=0*. When Tag_x receives the *Query* command issued from the *Reader*, Tag_x first generates a random number m and the current timestamp T_i. Then, Tag_x computes $M=(T_i||H(T_i)\__{Tag})\oplus H(m\oplus T\_key)$ and $C=H(m\oplus T\_key\oplus T_i)$ due to the *flag* value equals 1, where $H(T_i)\__{Tag}$ denotes the hash value of timestamp T_i is sent from Tag_x. In stead of applying binary string $(SID_i\oplus m)||T_i$ as in Condition 1, here we use the string $T_i||H(T_i)\__{Tag}$ to compute the outgoing response message M. This design can produce a randomized output to the current session even if the previous authentication session is not safely terminated. After dividing C into three blocks R_1, R_2 and R_3, Tag_x sends (R_1, M, m) as a response to the *Reader*.

3. ***Reader*** **→** ***Server*****:** ***R_1, M, m, S, R_value***

Reader sends an authentication request (R_1, M, m, S, R_value) to *Server*.

4. ***Server* → *Reader*: *N, n***
First, *Server* verifies the legitimacy of *Reader* with the same procedure in step 4 where *flag=0*. Upon receiving the authentication request, *Server* uses the EIM scheme to efficiently retrieve the corresponding data tuple in backend database. Then, *Server* computes $M \oplus H(m \oplus T\_key\_{DB})$ to obtain T_i and $H(T_i)\_{Tag}$. For the validity of the incoming message, *Server* verifies whether the computed value $H(T_i)$ is identical to $H(T_i)\_{Tag}$ (i.e., message integrity), and whether the appended timestamp is valid, $T_i > T_{i\_DB}$. If the verification processes are passed, *Server* will calculate $C' = H(m \oplus T\_key\_{DB} \oplus T_i)$ and divide C' into three blocks R'_1, R'_2 and R'_3. If R'_1 and R_1 are identical, *Server* will generate a random number n and compute $N = R'_2 \oplus H(n \oplus T\_key\_{DB})$. Finally, *Server* transmits (N, n) to *Reader* and updates $SID_{i+1\_DB} = H(T_i)\_{Tag} \oplus R'_3$ and $T_{i\_DB} = T_i$.
5. ***Reader* → *Tag*$_x$: *N, n***
Reader forwards the message (N, n) to Tag_x. Then, Tag_x computes $N \oplus H(n \oplus T\_key)$ to retrieve R'_2 and verifies the validity of incoming message (whether R'_2 equals R_2). If both values are the same, Tag_x changes the *flag* value from 1 to 0 and updates the value of security identification SID_{i+1} to $H(T_i)\_{Tag} \oplus R'_3$.

3.2 Efficient Identity Match Scheme

Each tag Tag_x stores four data in the memory including search seed ID_{tag}, update index k, current transaction number *TID,* and last successful transaction number *LST*. For each tag, the backend server maintains two records to prevent *DoS* attack (i.e., shared secret information is out of synchronization). Each record contains six fields in the database: (1) fast matching key $H^n(ID_{tag})$, (2) search index value n, (3) search seed ID_{tag}, (4) update threshold value k, (5) current transaction number *TID* and (6) last successful transaction number *LST*. Default values for the *TID* and *LST* are assigned to both the backend server and the corresponding tag during system initialization. For the sake of simplicity, the initial values of *TID* and *LST* are usually set as the same. The fast matching key is used as a search index to efficiently find the corresponding tuple from the backend database, where n is a pre-defined positive integer. For security enhancement, both the backend server and corresponding tag will update the search seed, ID_{tag}, when the current *TID* is equal to (or greater than) the update threshold value k'. The transaction numbers (*TID* and *LST*) also can be used to prevent replay attack [3]. The normal EIM process is shown in Figure 3. We describe the detailed procedure in the following.

1. ***Reader* → *Tag*$_x$: *Query***
Reader issues a *Query* signal to Tag_x.

2. ***Tag*$_x$ → *Reader* → *Server*: $H^m(ID_{tag})$*, m, TID,* Δ*TID***
After receiving the *Query* command, Tag_x generates random number m ($\leqq n$), and computes $H^m(ID_{tag})$, $TID=TID+1$, and $\Delta TID=TID-LST$. Then, Tag_x computes the hash chain value of ID_{tag} with m iterations (i.e., $H^m(ID_{tag})$) and sends the result with m, *TID*, ΔTID as a response message to *Reader*. *Reader* transmits an authentication request including $H^m(ID_{tag})$, m, *TID*, and ΔTID to *Server*.

3. *Server* → *Reader* → *Tag*$_x$: *Response*

When *Server* receives an authentication request sent from *Reader*, *Server* calculates $n - m$ times of one-way hash function iteratively with the received $H^m(ID_{tag})$ value as the starting seed to get the fast matching key $H^n(ID_{tag})=H^{n-m}(H^m(ID_{tag}))$. *Server* computes the current value of the transaction number TID' with equations $TID'= LST_{new} + \Delta TID$ or $TID'= LST_{old} + \Delta TID$, based on the corresponding tuple of the matching key $H^n(ID_{tag})$. In order to prevent the replay attack, if $TID' \neq TID$ or $TID' \leqq TID_{old}$, *Server* discards the incoming message. The transaction number is used to update the fast matching key for security enhancement in EIM scheme. Once the current transaction number is greater than or equal to the pre-defined update threshold value k, *Server* updates the search seed $ID_{tag\_new}=H(ID_{tag} \oplus k)$ and value k. With the new search seed ID_{tag_new}, the new fast matching key $I_{new} = H^n(ID_{tag\_new})$ is computed by *Server* for the data retrieval next time. Finally, *Server* updates the values of TID and LST ($TID_{new}=LST_{new}=TID'$, $TID_{old}=TID'$ and $LST_{old}=TID'-\Delta TID$), and sends a response to *Reader*. Upon receiving the response message forwarded from *Reader*, Tag_x updates the LST value with its TID value. Similarly, if TID is greater than or equal to the update threshold value k, then Tag_x updates k and search seed $ID_{tag} =H(ID_{tag} \oplus k)$. Note that the values of k and n, which can be any positive integers, are sensitive to the trade-off between system performance and security consideration. From the view of system performance, small n indicates less computation load for the tag and the backend server. In return, the tag anonymity would be vulnerable to break down. A small number k may result in computation burden at the backend server due to frequent updates of the fast matching key in each tuple. Nevertheless, a small number k can provide stronger forward security in the proposed EIM scheme.

4 Security Analysis

First of all, in our scheme only randomized message contents such as (R_1, M, N) and one time valid random numbers (i.e., m and n) are transmitted through the insecure communication channel. Therefore, message data security and tag anonymity property are guaranteed. Secondly, due to the verification process of timestamp of the last legitimate incoming message, this timestamp checking mechanism ensures our scheme against replay attack. Adversary cannot use the issued authentication message, already used in previous session, to iteratively attack the backend server. Furthermore, the forward security is embedded in the proposed scheme since the SID_i will be automatically updated after each successful authentication process. Finally, to resist *DoS* attack, we develop two processes (*flag*=0 and *flag*=1) to successfully update the SID_i even if the previous session is not safely terminated.

In terms of RFID system performance, we develop a novel process, called EIM scheme, to enhance both system performance and security while performing identity match process at the backend server. Our scheme utilizes the n iterative computation result of one-way hash function with the initial search seed ID_{tag} as the fast matching key for the corresponding entry table in the database. Instead of calculating the hashed value of secret identification for every entry to find the match entry in database, our scheme only needs to calculate $n - m$ times of one-way hash function iteratively with the received $H^m(ID_{tag})$ value as the starting seed. Besides, EIM scheme maintains two records, the last one and the current one, for each tag identity to defend against the *DoS* attack. In order to achieve anonymity property and forward security, a fast matching key updating mechanism with the update threshold value k is given at each tag. Replay attacks also can be solved by the proposed transaction number mechanism similar to [3]. Note that the decision of k and n depends on the design trade-off between system performance and security consideration as mentioned in Section 3.2 Finally, one of the most important contributions of EIM scheme is that EIM scheme can be implemented with any other published authentication scheme. In other words, EIM scheme is compatible with any hash-based authentication scheme for system performance enhancement.

Table 1 shows the proposed schemes are superior to the previous schemes by supporting all major security, privacy and system efficiency criterions in RFID applications environment.

Table 1. Comparison with previous authentication

	Data security	Anonymity	Resistance to replay attack	Resistance to DoS attack	Forward security	Backend server load
Weis et al. [1] (Hash-based access control)	X	X	X	O	X	Low
Weis et al. [1] (Randomized hash-locking access control)	X	O	X	O	X	High
Ohkubo et al. [2]	O	O	X	O	O	High
Henrici & Műller. [3]	O	X	O	O	X	Low
Yang et al. [6]	O	X	X	O	X	High
An & Oh [7]	X	O	X	O	X	High
Kim et al. [9]	X	X	X	O	O	High
Park & Lee [5]	O	X	X	O	X	Low
Osaka et al. [8]	O	X	X	X	X	High
Our Scheme	O	O	O	O	O	Low

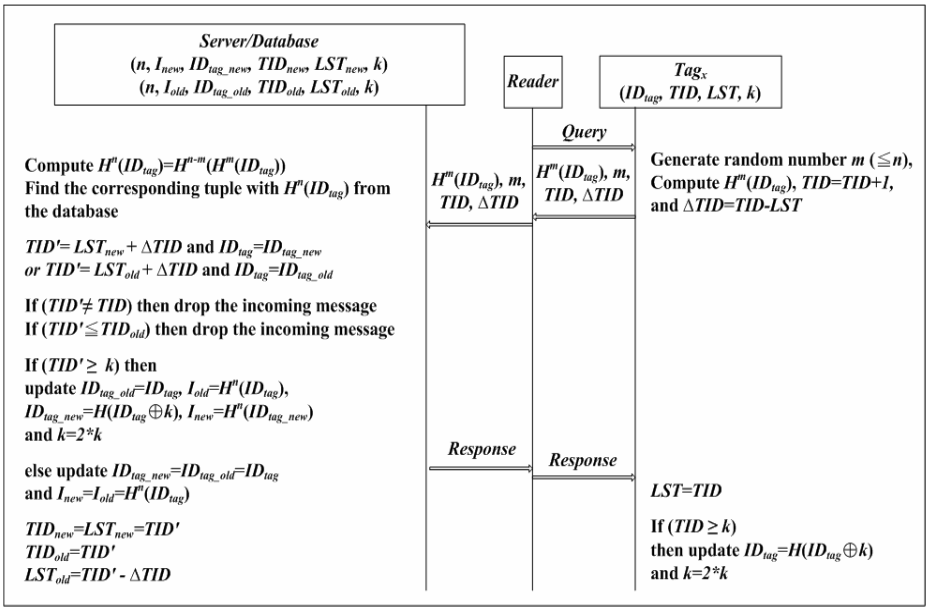

Server/Database
$(n, I_{new}, ID_{tag\_new}, TID_{new}, LST_{new}, k)$
$(n, I_{old}, ID_{tag\_old}, TID_{old}, LST_{old}, k)$
Reader
Tag_x
(ID_{tag}, TID, LST, k)
Query
Generate random number m ($\leq n$), Compute $H^m(ID_{tag})$, $TID=TID+1$, and $\Delta TID=TID-LST$
$H^m(ID_{tag})$, m, TID, ΔTID
$H^m(ID_{tag})$, m, TID, ΔTID
Compute $H^n(ID_{tag})=H^{n-m}(H^m(ID_{tag}))$
Find the corresponding tuple with $H^n(ID_{tag})$ from the database
$TID'=LST_{new}+\Delta TID$ and $ID_{tag}=ID_{tag\_new}$
or $TID'=LST_{old}+\Delta TID$ and $ID_{tag}=ID_{tag\_old}$
If ($TID' \neq TID$) then drop the incoming message
If ($TID' \leq TID_{old}$) then drop the incoming message
If ($TID' \geq k$) then
update $ID_{tag\_old}=ID_{tag}$, $I_{old}=H^n(ID_{tag})$,
$ID_{tag\_new}=H(ID_{tag} \oplus k)$, $I_{new}=H^n(ID_{tag\_new})$
and $k=2*k$
else update $ID_{tag\_new}=ID_{tag\_old}=ID_{tag}$
and $I_{new}=I_{old}=H^n(ID_{tag})$
$TID_{new}=LST_{new}=TID'$
$TID_{old}=TID'$
$LST_{old}=TID'-\Delta TID$
Response
Response
$LST=TID$
If ($TID \geq k$)
then update $ID_{tag}=H(ID_{tag} \oplus k)$
and $k=2*k$

Fig. 3. The proposed EIM

5 Conclusion

The main contribution of this study is to enhance previous proposed RFID authentication schemes and improve the efficiency of identity match mechanism at the backend server of a RFID system by developing a novel mutual authentication scheme and an efficient identity match scheme, respectively. Based on security analysis in Section 4, our RFID authentication scheme achieves the data security criterion, and the privacy requirements of tag anonymity and intractability. In addition, our schemes can defend against active attacks (*DoS* attack and replay attack) and provide better system performance on identity match process than other hash-based authentications. In conclusion, we have developed two useful and reliable schemes for RFID authentication operation to enhance data security, privacy protection and system performance in general RFID systems environment.

Acknowledgments. The authors gratefully acknowledge the support from *iCAST* and TWISC projects sponsored by the National Science Council, Taiwan, under the Grants No NSC95-3114-P-001-002-Y02, NSC95-3114-P-001-001-Y02 and NSC95-2218-E-011-015.

References

1. Weis, S.A., Sarma, S.E., Rivest, R.L., Engels, D.W.: Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems. In: Security in Pervasive Computing, pp. 201–212 (2003)
2. Ohkubo, M., Suzki, K., Kinoshita, S.: Cryptographic Approach to 'Privacyfriendly' Tags. In: RFID Privacy Workshop, MA, USA, MIT, Cambridge (2003)
3. Henrici, D., Müller, P.: Hash-based Enhancement of Location Privacy for Radio Frequency Identification Devices using Varying Identifiers. In: Workshop on Pervasive Computing and Communications Security (PerSec'04) at IEEE PerCom Workshop 2004, Orlando, Florida, March 14-17, 2004, IEEE, Los Alamitos (2004)
4. Yuksel, K.: Universal Hashing for Ultra-Low-Power Cryptographic Hardware Applications, Master Thesis, Dept. of Electronical Engineering, WPI (2004)
5. Park, J.-S., Lee, I.-Y.: RFID Authentication Protocol Using ID Synchronization in Insure Communication. In: The International Conference on Hybrid Information Technology (ICHIT'06), vol. 2, pp. 664–667 (2006)
6. Yang, J., Park, J., Lee, H., Ren, K., Kim, K.: Mutual Authentication Protocol for Low-cost RFID. In: The Encrypt Workshop on RFID and Lightweight Crypto (2005)
7. An, Y., Oh, S.: RFID System for User's Privacy Protection. In: Asia-Pacific Conference on Communications, pp. 516–519 (2005)
8. Osaka, K., Takagi, T., Yamazaki, K., Takahashi, O.: An Efficient and Secure RFID Security Method with Ownership Transfer. IEEE ICCIAS 2, 1090–1095 (2006)
9. Kim, H.-W., Lim, S.-Y., Lee, H.-J.: Symmetric Encryption in RFID Authentication Protocol for Strong Location Privacy and Forward-Security. In: the International Conference on Hybrid Information Technology (ICHIT'06), vol. 2, pp. 718–723 (2006)

Author Index