Lecture Notes in Computer Science 5645

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

Ehud Gudes Jaideep Vaidya (Eds.)

Data and Applications Security XXIII

23rd Annual IFIP WG 11.3 Working Conference
Montreal, Canada, July 12-15, 2009
Proceedings

Springer

Volume Editors

Ehud Gudes
Ben-Gurion University of the Negev, Department of Computer Science
Beer-Sheva, Israel, 84105
E-mail: ehud@cs.bgu.ac.il

Jaideep Vaidya
Rutgers University, MSIS Department
1 Washington Park, Newark, NJ 07102, USA
E-mail: jsvaidya@business.rutgers.edu

Library of Congress Control Number: 2009929941

CR Subject Classification (1998): E.3, D.4.6, C.2, F.2.1, J.1, K.6.5

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

ISSN 0302-9743
ISBN-10 3-642-03006-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-03006-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12718412 06/3180 5 4 3 2 1 0

Preface

This volume contains the papers presented at the 23rd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC) held at Concordia University, Montreal, Canada, July 12–15, 2009. This year's working conference continued its tradition of being a forum for disseminating original research results and practical experiences in data and applications security.

This year we had an excellent program consisting of 9 research paper sessions with 18 full research papers, and 4 short papers which were selected from a total of 47 submissions after a rigorous reviewing process by the Program Committee members and external reviewers. These sessions included such topics as access control, security policies, privacy, intrusion detection, trusted computing and data security in advanced application domains. In addition, the program included a keynote address, a tutorial and a panel session. We would like to thank Michael Reiter for his keynote address on "Better Architectures and New Security Applications for Coarse Network Monitoring." We would also like to thank Joachim Biskup for a stimulating tutorial on "How to protect Information: Inference Control for Logic-Oriented Information Systems."

The success of this conference was a result of the efforts of many people. We would like to extend our appreciation to the Program Committee members and external reviewers for their hard work. We would like to thank the General Chair, Mourad Debbabi, for taking care of the organizational aspects of the conference. We would also like to thank Claudio Ardagna for serving as the Publicity Chair and for promptly updating the conference website, Lingyu Wang for serving as the Local Arrangements Chair, and the IFIP WG 11.3 Chair, Vijay Atluri, for assisting us throughout. Special thanks go to Alfred Hofmann, Editorial Director at Springer, for agreeing to include these conference proceedings in the *Lecture Notes in Computer Science* series.

Last but not least, our thanks go to all of the authors who submitted papers and to all of the attendees. We hope you find the papers in this Volume stimulating and beneficial for your research.

July 2009

Ehud Gudes
Jaideep Vaidya

Conference Organization

Executive Committee

General Chair	Mourad Debbabi, Concordia University
Program Co-chairs	Ehud Gudes, Ben-Gurion University of the Negev Jaideep Vaidya, Rutgers University
Publicity Chair	Claudio Agostino Ardagna, University of Milan, Italy
Local Arrangements Chair	Lingyu Wang, Concordia University, Canada
IFIP WG 11.3 Chair	Vijay Atluri, Rutgers University

Program Committee

Gail-Joon Ahn	Arizona State University, USA
Claudio A. Ardagna	University of Milan, Italy
Vijay Atluri	Rutgers University, USA
Ken Barker	University of Calgary, Canada
Steve Barker	King's College London University, UK
Sabrina De Capitani di Vimercati	University of Milan, Italy
Soon Ae Chun	City University of New York, USA
Chris Clifton	Purdue University, USA
Jason Crampton	Royal Holloway, London University, UK
Mourad Debbabi	Concordia University, Canada
Steve Demurjian	University of Connecticut, USA
Wenliang Du	Syracuse University, USA
Yuval Elovici	Ben-Gurion University of the Negev, Israel
Csilla Farkas	University of South Carolina, USA
Eduardo B. Fernandez	Florida Atlantic University, USA
Elena Ferrari	University of Insubria, Italy
Qijun Gu	Texas State University, USA
Ehud Gudes	Ben-Gurion University, Israel
Murat Kantarcioglu	University of Texas at Dallas, USA
Peng Liu	Pennsylvania State University, USA
Sharad Mehrotra	University of California, Irvine, USA
Ravi Mukkamala	Old Dominion University, USA
Martin Olivier	University of Pretoria, South Africa
Sylvia Osborn	University of Western Ontario, Canada
Brajendra Panda	University of Arkansas, USA
Indrakshi Ray	Colorado State University, USA
Indrajit Ray	Colorado State University, USA

Pierangela Samarati	University of Milan, Italy
Andreas Schaad	SAP Research, Germany
Dongwan Shin	New Mexico Tech, USA
Basit Shafiq	Rutgers University, USA
Anoop Singhal	NIST, USA
Jaideep Vaidya	Rutgers University, USA
Lingyu Wang	Concordia University, Canada
Janice Warner	Georgian Court University, USA
Duminda Wijesekera	George Mason University, USA
Ting Yu	North Carolina State University, USA

Table of Contents

Access Control

Privacy II

Database Security II

Trusted Computing

Short Papers

Controlled Query Evaluation and Inference-Free View Updates

Joachim Biskup, Jens Seiler, and Torben Weibert

Technische Universität Dortmund, Dortmund, Germany
{biskup,seiler,weibert}@ls6.cs.uni-dortmund.de

Abstract. We extend Controlled Query Evaluation (CQE), an inference control method to enforce confidentiality in static information systems under queries, to updatable databases. Within the framework of the lying approach to CQE, we study user update requests that have to be translated into a new database state. In order to avoid dangerous inferences, some such updates have to be denied even though the new database instance would be compatible with a set of integrity constraints. In contrast, some other updates leading to an incompatible instance should not be denied. We design a control method to resolve this seemingly paradoxical situation and then prove that the general security definitions of CQE and other properties linked to user updates hold.

1 Introduction

The enforcement of confidentiality in information systems, that is allowing access to information only to authorised users, is often done by static mechanisms like *discretionary access control* or *mandatory access control*. While these mechanisms can prevent unauthorised users from a direct access to secret information such users might still be able to gather enough data to infer secrets indirectly. This is known as the *inference problem* [FJ02]. Inference channels can be created by various means, in particular by observing the system's behaviour linked to integrity constraints and other a priori knowledge about databases.

Controlled Query Evaluation (CQE) is a dynamic control method that counteracts such inferences. A comprehensive overview of CQE can be found in [BB04a]; see also [BW08]. All previous work on CQE deals with confidentiality in static databases. In this paper, we want to enable users of an information system to alter the contents of the database by means of issuing update requests, which should be appropriately reflected in the database.

Enabling users to update information is known to result in unwanted inferences as can be seen in *multilevel secure (MLS) databases* and the phenomenom of *polyinstantiation* (see, for example, [DA87, LD90, JS91, SJ92, CG99, CG01]). In MLS databases there exists data on high levels that is invisible to users on a lower level. If a low level user tries to insert data that does already exist on a higher level such an update request cannot be denied or otherwise the user would be able to infer the existence of that data on a higher, secret level. Therefore the update is accepted and the same data entity exists on two or more levels; the

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 1–16, 2009.

database becomes polyinstantiated. Controlled Query Evaluation shares certain similarities with polyinstantiated MLS databases in that CQE too has multiple levels. While the database administrator has a complete and exact knowledge of the database instance, the (unprivileged) database user constructs his own belief on the database instance using his a priori knowledge and the data that he has explicitly obtained from the system, namely the potentially modified answers.

We consider the a priori knowledge and the obtained answers to previous queries as the user's current *view* on the database instance. Thus we treat the data obtained similarly to the traditional database management, where a "view" is syntactically defined as an identified query, and semantically interpreted either dynamically or statically, by "materialising" the answer to the identified query. A classical problem is the question of how to translate an update request that refers to a view to the underlying database instance, in particular how to resolve ambiguities (see, for example [BS81, DB82, La90, He04, BP06]). In this paper, we will avoid the ambiguity problem by only allowing requests to change the truth value of atomic sentences.

However, we identify and solve the problem of how to process such requests in an inference-free way. More specifically, in Sect. 2, we summarize query processing under the lying form of CQE. Then, in Sect. 3, we increase the interaction of the user level with the administrator level by view updates. This feature adds new sources of possibly dangerous inferences that, however, we can hope to control by suitably combining a simple means with the concept of lying. In Sect. 4, we design a full update algorithm in detail, which complements inference-free query answering with inference-free view updating, and afterwards, in Sect. 5, we exemplify a run of this algorithm. Then, in Sect. 6, the important property of confidentiality is proven to hold. Finally, in Sect. 7, we show that any view update issued by the user can be undone by him.

2 Controlled Query Evaluation with Lying

We exploit a well-known logic-oriented approach to information systems, which provides formal semantics for both query answering and updating including enforcement of constraints. In particular, we assume the following prerequisites:

- A database *instance* db is a complete interpretation of a propositional logic (leaving generalisations to first-order logic [BB07] or to incomplete information systems [BW08] open for a future elaboration), such that a sentence of a propositional logic is either *true* or *false* with respect to a database instance but never undefined; an instance can be specified by listing for each atomic proposition χ whether that proposition or its negation should be included.
- A *query* ϕ is a (closed) sentence in the logic (suggesting again a generalisation to open queries [BB07] for future work).
- Accordingly, a query ϕ is *evaluated* relative to an instance db by the function $eval(\phi)(db)$ returning the pertinent truth value or, equivalently, either ϕ or $\neg\phi$, which is denoted by $eval^*(\phi)(db)$.
- We employ the usual notion of logical *implication*, denoted by $\models$.

Furthermore, we deal with inference control in a specific form of Controlled Query Evaluation that modifies a potentially harmful answer to a query by *lying* [BK95, BB01, BW08], that is returning the negation of the true answer (leaving the alternative forms of *refusal* [SJ83, BB01, BW08], that is returning *mum* instead of the true answer, and a *combined* form [BB04b] open for further research). The lying approach is based on the following additional prerequisites:

- The *confidentiality policy*, supposed to be *known* to the user, is declared as a set of potential secrets, that is a finite set *pot_sec* of sentences to be protected so that the user can never infer that any sentence $\psi \in pot\_sec$ actually holds.
- In order to preserve this kind of confidentiality, we have to protect not only the individual potential secrets but, in fact, the disjunction of all potential secrets $pot\_sec\_disj := \bigvee_{\psi \in pot\_sec} \psi$. This is necessary to avoid "hopeless situations" which lead to an inconsistency (see [BB01, BB04a] for further explanation).
- Basically, the control mechanism maintains a *user log log* for keeping the (postulated) a priori knowledge of the user and the answers returned to previous queries, and invokes a *censor* $censor^L$ for inspecting whether in a given situation the true answer $eval^*(\phi)(db)$ to a query ϕ will be harmful; more formally the function $censor^L$ returns a Boolean value as follows:

$$censor^L(db, pot\_sec, log, \phi) := log \cup \{eval^*(\phi)(db)\} \models pot\_sec\_disj. \tag{1}$$

- CQE (with uniform lying) is then a function $control\_eval(Q, log_0)(db, pot\_sec)$ where Q is any finite *sequence of (closed) queries* $Q = \langle \phi_1, \phi_2, \ldots, \phi_n \rangle$, log_0 is an initial *user log* representing the a priori *user knowledge*, *db* is a *database instance* and *pot_sec* is the *confidentiality policy* to be enforced. The function *control_eval* returns a corresponding sequence of answers and updated user logs (ans_i, log_i). The return values are calculated by censoring the true answer of a user's query ϕ_i and in case of unwanted inferences by applying a *modification* (by lying) to the answer:

$$\begin{aligned} ans_i := \ & \text{if } censor^L(db, pot\_sec, log_{i-1}, \phi_i) \text{ then } \neg eval^*(\phi_i)(db) \\ & \text{else } eval^*(\phi_i)(db) \end{aligned} \tag{2}$$

$$log_i := log_{i-1} \cup \{ans_i\} \tag{3}$$

It was shown in [BB01] that the confidentiality property expressed by the following definition holds for this variant of CQE.

Definition 1 (Confidentiality). *A function* $control\_eval(Q, log_0)(db, pot\_sec)$ preserves confidentiality *iff for all finite sequences of queries* Q*, all initial user knowledges* log_0*, all instances db satisfying* log_0*, all sets of potential secrets pot_sec and all potential secrets* $\psi \in pot\_sec$ *with* $log_0 \not\models \psi$ *there exists an instance* db^S *satisfying* log_0 *such that*

1. *db and* db^S *return the same sequence of answers:*
   $control\_eval(Q, log_0)(db, pot\_sec) = control\_eval(Q, log_0)(db^S, pot\_sec)$
2. db^S *does not contain the potential secret* ψ*:* $eval^*(\psi)(db^S) = \neg\psi$

3 Inferences through View Updates

Before formally introducing a model for view updates under CQE, we take an informal look at user modifications which demonstrate the necessity for a non-trivial means of handling a user's update requests. Simply allowing all "accepted" updates or forbidding all "denied" updates leads to inferences as the following examples show.

Example 1 (Inferences through accepted update requests). Suppose, a user issues an update request on a database restricted by an integrity constraint, e.g., $a \vee b$. He wants to ensure that the interpretation of a is *true*. If the system responds with "Value of a changed to *true*", the user can infer that previously a was *false* and that then b must have been and still is *true*.

If b is a secret to be protected then clearly the update request would have had to be denied. Even if the truth of b is not to be kept secret, the system has in some way to keep track of the user's ability to infer the truth of b in order to prevent future inferences.

Example 2 (Inferences through denied update requests). Suppose, a database instance $\{\neg a, b\}$ underlies the integrity constraint $\neg a \vee \neg b$. Again a user issues a request to ensure the truth of a. This, however, would lead to an integrity violating instance $\{a, b\}$. Denying the update request enables the user to reason about the value of b: if b was *false*, then the value of a would not matter. But since the update request was denied, b must in fact be *true*.

Again the user is able to infer the truth value of a variable he doesn't directly change or query. Interestingly enough, such inferences can be easily calculated in advance and thus be encountered. The necessary tool for that is a simple negation of variables in the formulas of propositional logic, as described next.

We define the set *FORMULA* of allowed propositional formulas in the usual inductive way: *true*, *false* and each $v \in VAR$ are elements of *FORMULA*; if t_1 and t_2 are elements of *FORMULA*, then also $\neg t_1$, $(t_1 \wedge t_2)$ and $(t_1 \vee t_2)$. We will omit brackets where not necessary, and omit double negation where convenient.

While a variable serves as a placeholder for a truth value we will use literals as a means to specify such a truth value for a given variable:

Definition 2 (Literal). *For every variable $v \in VAR$, v and $\neg v$ are literals of the set LIT. For a literal $\chi \in \{v, \neg v\}$ with $v \in VAR$, we define $\chi^+ := v$. That way the symbol χ^+ is syntactically identical to the variable whose truth value is being defined by the literal χ.*

We can now define the negation of variables on formulas. Although we negate variables we will specify the variable to be negated by a literal. This is done to simplify the usage of variable negation later on when a user specifies the updated truth value of a variable using a literal.

Definition 3 (Negation of variables on formulas). *The negation of variables on formulas $neg(\cdot,\cdot) : FORMULA \times LIT \longrightarrow FORMULA$ is defined by*

$$neg(\phi, \chi) := \begin{cases} \phi & \text{for } \phi \in \{false, true\} \\ \phi & \text{for } \phi = v \neq \chi^+, v \in VAR \\ \neg\phi & \text{for } \phi = \chi^+ \\ \neg\big(neg(\phi', \chi)\big) & \text{for } \phi = \neg\phi' \\ \big(neg(\phi', \chi) \wedge neg(\phi'', \chi)\big) & \text{for } \phi = \phi' \wedge \phi'' \\ \big(neg(\phi', \chi) \vee neg(\phi'', \chi)\big) & \text{for } \phi = \phi' \vee \phi'' \end{cases} \tag{4}$$

Informally, every appearance of a variable specified by a literal χ in the formula ϕ is negated. We call $neg(\phi, \chi)$ the χ-negated formula ϕ. Likewise, the negation of variables on a set of formulas $neg(\cdot, \cdot) : \mathcal{P}(FORMULA) \times LIT \longrightarrow \mathcal{P}(FORMULA)$ is defined by

$$neg(M, \chi) := \{\ neg(\phi, \chi)\ \mid\ \phi \in M\ \}. \tag{5}$$

Example 3 (Negation of variables on a formula).

$$\begin{aligned} neg\big(\neg(a \wedge b) \vee \neg a, \neg a\big) &= neg\big(\neg(a \wedge b), \neg a\big) \vee neg(\neg a, \neg a) && (6)\\ &= \neg\big(neg(a \wedge b, \neg a)\big) \vee \neg\big(neg(a, \neg a)\big) && (7)\\ &= \neg\big(neg(a, \neg a) \wedge neg(b, \neg a)\big) \vee \neg(\neg a) && (8)\\ &= \neg(\neg a \wedge b) \vee a && (9) \end{aligned}$$

We can now identify a simple property of variable negation as defined above:

Lemma 1 (Negation equivalence). *For any instance db, all $\phi \in FORMULA$, any literal χ, using the following definition of db^χ via db,*

$$db^\chi := \begin{cases} \big(db \setminus \{\chi\}\big) \cup \{\neg\chi\} & \text{for } \chi \in db \\ \big(db \setminus \{\neg\chi\}\big) \cup \{\chi\} & \text{otherwise} \end{cases} \tag{10}$$

we have that

$$eval(\phi)(db) = eval\big(neg(\phi, \chi)\big)\big(db^\chi\big). \tag{11}$$

This means that we get the same results evaluating a formula on an instance and evaluating the χ-negated formula on the instance created by negating the variable specified by χ.

Lemma 1 can now be used to identify inferences as they appeared in Examples 1 and 2. We claim (and prove below) that

- the set $neg(log, \chi)$ contains valid formulas *after* changing the truth value of χ, if beforehand the formulas contained in *log* were true, and
- the formula $neg\big(\neg(\bigwedge_{\phi \in constraints} \phi), \chi\big)$ describes the knowledge gained by learning that the set of *constraints* doesn't hold under an interpretation where the truth value of the variable specified by χ has been negated, provided that the *constraints* were valid beforehand.

This can be utilised to create a secure, that is inference-free, view update mechanism for Controlled Query Evaluation under a uniform lying censor.

4 Inference-Free View Updates under CQE with Lying

Definition 4 (Controlled Query Evaluation with view updates). *We define a sequence Q of queries* and update requests *by:*

$$Q := \langle\ \Theta_1, \Theta_2, \ldots, \Theta_i, \ldots \Theta_k\ \rangle \quad \textit{with} \tag{12}$$

$$\Theta_i := \begin{cases} \Phi_i & \textit{a query with } \Phi_i \in \textit{FORMULA or} \\ \textit{update}(\chi_i) & \textit{a (view) update operation with } \chi_i \in \textit{LIT} \end{cases} \tag{13}$$

Additionally we have the following:

- $\textit{constraints} \subseteq \mathcal{P}(\textit{FORMULA})$ *is a finite set of constraints, which have to be satisfied before and after each update,*
- $\textit{log}_0 \subseteq \mathcal{P}(\textit{FORMULA})$ *is an initial set of the assumed user knowledge with* $\textit{log}_0 \supseteq \textit{constraints}$*,*
- $\textit{db}_0$ *is an initial database instance and*
- $\textit{pot\_sec} \subseteq \mathcal{P}(\textit{FORMULA})$ *is a finite set of potential secrets.*

Then we define Controlled Query Evaluation with view updates by

$$\textit{control\_eval\_update}(Q, \textit{log}_0, \textit{constraints})(\textit{db}_0, \textit{pot\_sec}) \tag{14}$$

$$= \big\langle (\textit{ans}_1, \textit{log}_1, \textit{db}_1), \ldots, (\textit{ans}_i, \textit{log}_i, \textit{db}_i), \ldots (\textit{ans}_k, \textit{log}_k, \textit{db}_k) \big\rangle \tag{15}$$

For queries Φ_i, we define the triple $(\textit{ans}_i, \textit{log}_i, \textit{db}_i)$ as in normal CQE with $\textit{db}_i := \textit{db}_{i-1}$. Update requests $\textit{update}(\chi_i)$ are defined by the algorithm described in the following and formalised in Def. 5.

Using the properties of *neg* as identified in Sect. 3 we can describe an algorithm that provides for inference-free view updates. The algorithm consists of four steps which also represent four disjunct cases determining the response to the user. These cases can be outlined roughly as follows:

1. The requested update is already compatible to the user's view and thus the database instance is not to be modified.
2. Allowing the requested update would infer a secret or be incompatible with the set of constraints and this fact is known to the user a priori.
3. Allowing the requested update would be incompatible with the set of constraints and this is unknown to the user a priori.
4. The requested update is accepted and the user receives confirmation.

In the following, we shall describe each of the cases in some more detail.

1. The first case is comparable with the property of acceptability in Def. 3.1 in [BS81], where view updates without confidentiality requirements are studied. An update that is already compatible with the current view needs not to be performed. In our case, a user's request to update χ_i is compatible with his current view if we have that

$$\textit{control\_eval}\big(\langle \chi_i \rangle, \textit{log}_{i-1}\big)\big(\textit{db}_{i-1}, \textit{pot\_sec}\big) = \langle (\chi_i, \textit{log}_{i-1} \cup \{\chi_i\}) \rangle. \tag{16}$$

Under the lying censor as defined in Sect. 2 this is equivalent to:

$$eval^*(\chi_i)(db_{i-1}) = \chi_i \quad AND\ log_{i-1} \cup \{\chi_i\} \not\models pot\_sec\_disj \tag{17}$$

$$OR\ eval^*(\chi_i)(db_{i-1}) = \neg\chi_i\ AND\ log_{i-1} \cup \{\neg\chi_i\} \models pot\_sec\_disj \tag{18}$$

So, if that condition holds we will tell the user that χ is already valid and update the log accordingly.

2. If the first case did not occur, then obviously $\neg\chi$ is valid from the user's point of view and a view update can take place if confidentiality or consistency isn't threatened. To verify this we check if the updated view would imply the disjunction of all potential secrets. We claim that this is the case if the following condition holds:

$$neg(log_{i-1}, \chi_i) \cup \{\chi_i\} \cup constraints \models pot\_sec\_disj \tag{19}$$

Interestingly we don't use the actual instance db_{i-1} and thus the aforementioned condition can be evaluated purely from information available to the user. Consequently the only additional information learned by the user is the passing of the first case and thus the fact that $\neg\chi_i$ is true within his view. This fact is added to the log.

3. While the previous case takes care of updates leading to inconsistencies of which the user is aware himself, the introducing Example 2 shows that there are also cases in which a user doesn't know that his update would lead to an inconsistent instance and additionally where the user *isn't allowed to know* that this would be the case.
We introduce $con\_conj := \bigwedge_{\phi \in constraints} \phi$ as the conjunction of all constraints and can easily verify if the future instance would be inconsistent after the user's update:

$$eval\big(con\_conj\big)\big((db_{i-1} \setminus \{\neg\chi_i\}) \cup \{\chi_i\}\big) = false \tag{20}$$

Additionally we have to check if telling the user about such an inconsistency would enable him to infer the disjunction of the potential secrets. Again we can make use of the properties of *neg* and require the following condition to be true in order to tell the user about an inconsistency found by the previous condition (20):

$$log_{i-1} \cup \{\neg\chi_i\} \cup \big\{neg(\neg con\_conj, \chi_i)\big\} \not\models pot\_sec\_disj \tag{21}$$

If both (20) and (21) are true then the user is informed about the inconsistency and his update request is denied. Additionally the log has to be updated by adding $\{\neg\chi_i\} \cup \big\{neg(\neg con\_conj, \chi_i)\big\}$.

4. This last case automatically occurs if all conditions of the three cases before do not hold. From the user's point of view the update is to be accepted now and consequently this fourth case notifies the user about his successful update. The log is being updated with the premise of condition (19) from the second case and we have $log_i := neg(log_{i-1}, \chi_i) \cup \{\chi_i\} \cup constraints \not\models pot\_sec\_disj$.

However, as we learned from the third case there are updates that would be inconsistent with the set of constraints of the database scheme but which are nevertheless to be allowed in order to protect secrets. Therefore, in case condition (20) was indeed true but the protecting condition (21) was false, we won't update the actual database instance making the fact that we tell the user about a successful update a lie.

As we see from the description of the four cases, the proposed algorithm employs lies at two places: Firstly, it can lie about the current view of the user before the update and tell him either that his requested update is already compatible with the database instance although it isn't (then (18) is true) or the database instance already contains the literal as desired by the requested update but telling so would implicate a secret or inconsistency (and thus condition (17) is *not* true). Secondly, we may lie to the user about the affirmation of a successful update despite not touching the actual instance in the fourth case.

Given the brief outline and the more detailed descriptions, we are now ready to declare the algorithm formally.

Definition 5 (A secure view update algorithm for CQE)
if (condition for case 1)

$$eval^*(\chi_i)(db_{i-1}) = \chi_i \quad AND\ log_{i-1} \cup \{\chi_i\} \not\models pot\_sec\_disj \tag{22}$$

$$OR\ eval^*(\chi_i)(db_{i-1}) = \neg\chi_i\ AND\ log_{i-1} \cup \{\neg\chi_i\} \models pot\_sec\_disj \tag{23}$$

then

- $db_i := db_{i-1}$, $log_i := log_{i-1} \cup \{\chi_i\}$
- $ans_i :=$ *"The requested update is already contained in the database"*

else if (condition for case 2)

$$neg(log_{i-1}, \chi_i) \cup \{\chi_i\} \cup constraints \models pot\_sec\_disj \tag{24}$$

then

- $db_i := db_{i-1}$, $log_i := log_{i-1} \cup \{\neg\chi_i\}$
- $ans_i :=$ *"Updating χ_i is inconsistent with secrets or integrity"*

else if (condition for case 3)

$$eval\big(con\_conj\big)\big((db_{i-1} \setminus \{\neg\chi_i\}) \cup \{\chi_i\}\big) = false \tag{25}$$

$$AND\ log_{i-1} \cup \{\neg\chi_i\} \cup \big\{neg(\neg con\_conj, \chi_i)\big\} \not\models pot\_sec\_disj \tag{26}$$

then

- $db_i := db_{i-1}$, $log_i := log_{i-1} \cup \{\neg\chi_i\} \cup \big\{neg(\neg con\_conj, \chi_i)\big\}$
- $ans_i :=$ *"Updating χ_i is incompatible with integrity"*

else (case 4)

- *if condition* (25) *then* $db_i := db_{i-1}$ *else* $db_i := (db_{i-1} \setminus \{\neg\chi_i\}) \cup \{\chi_i\}$
- $log_i := neg(log_{i-1}, \chi_i) \cup \{\chi_i\} \cup constraints$
- $ans_i :=$ *"Update of χ_i successful"*

5 An Example

To illustrate the algorithm of Def. 5 we give an example that will trigger all of the algorithm's cases:

- $pot\_sec := \{s_1, s_2\}$
- $constraints := \{a \Rightarrow s_1, c \Rightarrow b, s_2 \Rightarrow \neg c\}$
- $db_0 := \{a, \neg b, \neg c, s_1, s_2\}$
- $log_0 := \{a \Rightarrow s_1, c \Rightarrow b, s_2 \Rightarrow \neg c\}$
- $Q := \langle update(\neg a), update(c), update(b), update(c), update(a), update(b)\rangle$

1. $update(\neg a)$ will trigger case 1 due to condition (23):

$$eval^*(\neg a)(\{a, \neg b, \neg c, s_1, s_2\}) = a \; AND \; \{a \Rightarrow \; s_1, c \Rightarrow b, s_2 \Rightarrow \neg c\} \cup \{a\} \models s_1 \vee s_2$$

The algorithm has to lie in order to protect the disjunction of secrets. Despite a being true under db_0 it has to tell the user that $\neg a$ is already true since otherwise the user could infer that a was valid which would have implied the truth of the secret s_1. We thus get:

- $db_1 := \{a, \neg b, \neg c, s_1, s_2\}$
- $log_1 := \{a \Rightarrow s_1, c \Rightarrow b, s_2 \Rightarrow \neg c, \neg a\}$
- $ans_1 :=$ "The requested update is already contained in the database"

2. $update(c)$ will trigger case 3 since we have firstly, that an inconsistent database instance would be created (equation (25) holds):

$$eval\big(\{a \Rightarrow s_1 \wedge c \Rightarrow b \wedge s_2 \Rightarrow \neg c\}\big)\big((\{a, \neg b, \neg c, s_1, s_2\} \setminus \{\neg c\}) \cup \{c\}\big) = false$$

and secondly, that this can be told to the user without implying a secret (equation (26) holds):

$$\{a \Rightarrow s_1, c \Rightarrow b, s_2 \Rightarrow \neg c, \neg a\} \cup \{\neg c\} \cup \big\{neg(\neg con\_conj, c)\big\} \not\models \; pot\_sec\_disj$$

This is because with $\{\neg a, \neg b, \neg c, \neg s_1, \neg s_2\}$ we have a "witness" instance that makes true the premise of the implication but falsifies the conclusion. The same witness instance can be used to verify that indeed cases 1 and 2 will not be triggered. We now have:

- $db_2 := \{a, \neg b, \neg c, s_1, s_2\}$
- $log_2 := \{a \Rightarrow s_1, c \Rightarrow b, s_2 \Rightarrow \neg c, \neg a\} \cup \{\neg c\} \cup neg\big(\neg(a \Rightarrow s_1 \wedge c \Rightarrow b \wedge s_2 \Rightarrow \neg c), c\big)$
  $= \{a \Rightarrow s_1, c \Rightarrow b, s_2 \Rightarrow \neg c, \neg a, \neg c, \neg(a \Rightarrow s_1 \wedge \neg c \Rightarrow b \wedge s_2 \Rightarrow c)\}$
- $ans_2 :=$ "Updating c is incompatible with integrity"

3. $update(b)$ will trigger case 4 and modify the instance since neither is case 1 triggered (the same "witness" $\{\neg a, \neg b, \neg c, \neg s_1, \neg s_2\}$ can be used to verify this) nor is case 2 triggered which can be verified using the witness instance $\{\neg a, b, \neg c, \neg s_1, \neg s_2\}$. We also have that case 3 isn't triggered because $eval(con\_conj)(\{a, b, \neg c, s_1, s_2\}) = true$. Therefore a true update of the instance is done and we get:

- $db_3 := \{a, b, \neg c, s_1, s_2\}$
- $log_3 := neg\big(\{a \Rightarrow s_1, c \Rightarrow b, s_2 \Rightarrow \neg c, \neg a, \neg c, \neg(a \Rightarrow s_1 \wedge \neg c \Rightarrow b \wedge s_2 \Rightarrow c)\}, b\big) \cup \{b\} \cup \mathit{constraints}$
  $= \{a \Rightarrow s_1, c \Rightarrow \neg b, s_2 \Rightarrow \neg c, \neg a, \neg c, \neg(a \Rightarrow s_1 \wedge \neg c \Rightarrow \neg b \wedge s_2 \Rightarrow c), b, c \Rightarrow b\}$
- $ans_3 :=$ "Update of b successful"

For those readers not willing to solve the logic puzzle of the now quite stuffed log we hint that only four instances remain possible: $\{\neg a, b, \neg c, \neg s_1, \neg s_2\}$, $\{\neg a, b, \neg c, \neg s_1, s_2\}$, $\{\neg a, b, \neg c, s_1, \neg s_2\}$ and $\{\neg a, b, \neg c, s_1, s_2\}$. As such the user has gained a complete knowledge about the non-secrets a, b and c while it appears possible to him that all secrets are false.

4. $update(c)$ will trigger case 4 but not modify the instance. Again cases 1 and 2 are not triggered as the reader can easily verify. This time, however, we have that a real update would create an instance not compatible with the set of constraints. The instance $\{a, b, c, s_1, s_2\}$ violates the constraint $s_2 \Rightarrow \neg c$. However, telling this to the user would imply the truth of s_2 since from the users point of view the other two constraints cannot be violated. Luckily our condition (26) of the update algorithm protects us from relating a violation of constraints to the user by remaining "silent" and thus telling him the lie of a successful update:

- $db_4 := \{a, b, \neg c, s_1, s_2\}$
- $log_4 := neg\big(\{a \Rightarrow s_1, c \Rightarrow \neg b, s_2 \Rightarrow \neg c, \neg a, \neg c, \neg(a \Rightarrow s_1 \wedge \neg c \Rightarrow \neg b \wedge s_2 \Rightarrow c), b, c \Rightarrow b\}, c\big) \cup \{c\} \cup \mathit{constraints}$
  $= \{a \Rightarrow s_1, \neg c \Rightarrow \neg b, s_2 \Rightarrow c, \neg a, c, \neg(a \Rightarrow s_1 \wedge c \Rightarrow \neg b \wedge s_2 \Rightarrow \neg c), b, \neg c \Rightarrow b, c \Rightarrow b, s_2 \Rightarrow \neg c$
- $ans_4 :=$ "Update of c successful"

With this update the set of possible instances is reduced to two, namely $\{\neg a, b, c, \neg s_1, \neg s_2\}$ and $\{\neg a, b, c, s_1, \neg s_2\}$.

5. $update(a)$ will trigger case 2 since case 1 isn't triggered and an update of a would imply the secret s_1 which is captured by condition (24) of the algorithm. We now have:

- $db_5 := db_4$
- $log_5 := log_4$ (since $\neg a$ was already in log_4)
- $ans_5 :=$ "Updating a is inconsistent with secrets or integrity"

6. $update(b)$ triggers case 1 via condition (22). We get:

- $db_6 := db_5$
- $log_6 := log_5$ (since b was already in log_5)
- $ans_6 :=$ "The requested update is already contained in the database"

The example shows that every case of the algorithm is reachable and that cases 1 and 4 can both be lying or telling the truth. Additionally it visualises the fact that from the user's point of view there does always exist at least one instance with all secrets false that is consistent with the previous answers given by the update algorithm. In our case these are $\{\neg a, \neg b, \neg c, \neg s_1, \neg s_2\}$ for the instance before the first three updates, $\{\neg a, b, \neg c, \neg s_1, \neg s_2\}$ for the instance after the update of b and $\{\neg a, b, c, \neg s_1, \neg s_2\}$ for the updates after that.

6 Properties of Secure View Updates

Given the algorithm from Def. 5, the following lemma states that the fundamental invariant of Controlled Query Evaluation under lying (see, for example, [BK95, BB01, BW08]) applies to secure view updates, too.

Lemma 2 (Invariant). *For any instance of Controlled Query Evaluation with view updates control_eval_update with a sequence* $Q = \langle \Theta_1, \Theta_2, \ldots, \Theta_i, \ldots \Theta_k \rangle$ *the following invariant for the user log* log_i *holds:* $log_i \not\models pot\_sec\_disj$.

Proof. For queries $\Theta_i = \Phi_i$, the claim follows directly from the properties of normal CQE as shown in [BB01]. For updates $\Theta_i = update(\chi_i)$, we have to argue about the four different possible cases and their log updates:

1. If the update algorithm responds with an answer "The requested update is already contained in the database" then we have that either condition (17) or (18) is true. In the first case we directly get $log_i = log_{i-1} \cup \{\chi_i\} \not\models pot\_sec\_disj$. From the second case we get $log_{i-1} \cup \{\neg\chi_i\} \models pot\_sec\_disj$ and by induction we also have that $log_{i-1} \not\models pot\_sec\_disj$. From both follows $log_i = log_{i-1} \cup \{\chi_i\} \not\models pot\_sec\_disj$.
2. If the update algorithm responds with "Updating χ_i is inconsistent with secrets or integrity" we obviously have neither (17) nor (18) to be true:

$$NOT \Big(\quad (eval^*(\chi_i)(db_{i-1}) = \chi_i \quad AND\ log_{i-1} \cup \{\chi_i\} \not\models pot\_sec\_disj)$$
$$OR\ (eval^*(\chi_i)(db_{i-1}) = \neg\chi_i\ AND\ log_{i-1} \cup \{\neg\chi_i\} \models pot\_sec\_disj) \Big)$$

Using the laws of de Morgan and the completeness of the database instance we obtain the following equivalent expression:

$$(\underbrace{eval^*(\chi_i)(db_{i-1}) = \neg\chi_i}_{a}\ OR\ \underbrace{log_{i-1} \cup \{\chi_i\} \models pot\_sec\_disj}_{b})$$
$$AND\ (\underbrace{eval^*(\chi_i)(db_{i-1}) = \chi_i}_{c}\ OR\ \underbrace{log_{i-1} \cup \{\neg\chi_i\} \not\models pot\_sec\_disj}_{d})$$

Thus we have that (a *OR* b) *AND* (c *OR* d) must be true and it follows from the completeness of the database instance that exactly one of a or c can be true. Simply enumerating all combinations of a, b, c and d show that only three such combinations are possible: for $\neg a, b, c, \neg d$ it would follow that

$$log_{i-1} \cup \{\chi_i\} \models pot\_sec\_disj\ AND\ NOT\ (log_{i-1} \cup \{\neg\chi_i\} \not\models pot\_sec\_disj)$$

and thus

$$log_{i-1} \cup \{\chi_i\} \models pot\_sec\_disj\ AND\ log_{i-1} \cup \{\neg\chi_i\} \models pot\_sec\_disj$$

holds. This however is a contradiction to the induction hypothesis and thus this case cannot occur.
From the two other combinations, that is $\neg a, b, c, d$ and $a, \neg b, \neg c, d$ it follows directly from d that $log_i = log_{i-1} \cup \{\neg\chi_i\} \not\models pot\_sec\_disj$ is true.

3. The third case of the algorithm, the response "Updating χ_i is incompatible with integrity", can only occur if condition (21) holds. Therefore we directly have that

$$log_i = log_{i-1} \cup \{\neg\chi_i\} \cup neg(\neg con\_conj, \chi_i) \not\models pot\_sec\_disj$$

4. Finally, if none of the previous cases were triggered, we have that the update is accepted. From the non-occurrence of the second case we have that condition (19) does not hold, from which we directly obtain that

$$neg(log_{i-1}, \chi_i) \cup \{\chi_i\} \cup constraints \not\models pot\_sec\_disj$$

is true, which shows that the log update of fourth case (the last case) is also consistent with the invariant we want to show.

From the invariant $log_i \not\models pot\_sec\_disj$ it follows that the log does not imply any potential secret. This, however, does not mean that the *answers* of the update algorithm will never enable a reasoning user to infer the truth of a secret. In order to show that our claims from the end of Sect. 3 are correct and their usage in the update algorithm provides security we will define Controlled Query Evaluation with view updates to be secure if a user can never infer that any particular potential secret is true in the actual database instance. This requirement is captured by demanding that there always exists an alternative database instance in which the respective potential secret is false, but under which the same answers are returned as under the actual database instance. We therefore adapt Def. 1 so it is compatible with the sequence of produced databases under view updates.

Definition 6 (Confidentiality for view updates). *We say that a function* $control\_eval\_update(Q, log_0, constraints)(db_0, pot\_sec)$ *preserves confidentiality iff for all sequences of queries and update requests* Q*, all initial user knowledges* log_0*, all sets of constraints* $constraints \subseteq log_0$*, all instances* db_0 *satisfying* log_0*, all sets of potential secrets* pot_sec*, all potential secrets* $\psi \in pot\_sec$ *with* $log_0 \not\models \psi$ *there exists an instance* db_0^S *satisfying* log_0*, such that*

1. db_0 *and* db_0^S *return the same sequence of answers:*

$$\upsilon\big(control\_eval\_update(Q, log_0, constraints)(db_0, pot\_sec)\big) = \upsilon\big(control\_eval\_update(Q, log_0, constraints)(db_0^S, pot\_sec)\big)$$

2. db_0^S *does not contain the secret* ψ*:* $\quad eval^*(\psi)(db_0^S) = \neg\psi$

Above, we define υ *to be the projection of a set of tuples of the form* (ans_i, log_i, db_i) *to the user-visible set of answers* ans_i*.*

Theorem 1. *CQE with secure view updates, i.e., the function* $control\_eval\_update(Q, log_0, constraints)(db_0, pot\_sec)$ *as defined by Def. 4 together with Def. 5, preserves confidentiality in the sense of Def. 6.*

To prove this theorem, we need the following lemma, which can easily be proven via a backwards induction, which we omit here due to the lack of space.

Lemma 3. *For any sequence Q with length k there exists a sequence of instances db_i^S with the following two properties (where model_of means "makes true"):*

$$db_{i-1}^S := \begin{cases} (db_i^S \setminus \{\chi_i\}) \cup \{\neg\chi_i\} & \textit{for an update triggering case 4} \quad (27) \\ db_i^S & \textit{otherwise (queries or cases 1 to 3)} \quad (28) \end{cases}$$

$$db_i^S \ \textit{model\_of}\ log_i \cup \{\neg \textit{pot\_sec\_disj}\} \quad (29)$$

Proof (of Theorem 1). We show via induction that our system creates the same sequence of answers for a given sequence Q regardless if it started on db_0 or on db_0^S. This also means that the same sequence of logs is created.

For $i = 0$, we have $log_0 = log_0^S$ and no answers so far. For the step from $i-1$ to i, we differentiate between the nature of the operation $\Theta_i \in Q$. If Θ_i is a query Φ_i, then it follows from Lemma 3 and the proofs of confidentiality for CQE in terms of Def. 1 that the answer returned is the same under db and db^S.

If on the other hand Θ_i is an update $update(\chi_i)$, then we enumerate over the four possible cases of that operation under the instance db_i and show the identical reaction on the corresponding instance db_i^S:

1. From case 1 under db we have that $log_i := log_{i-1} \cup \{\chi_i\}$ and via Lemma 3 it follows that $db_i^S \ \textit{model\_of}\ log_i \subseteq \{\chi_i\}$. Assuming case 1 under db and construction of db_{i-1}^S via (28) we have that $eval^*(db_{i-1}^S)(\chi_i) = \chi_i$ which satisfies the first part of equation (22). We show that the second part, that is $log_{i-1}^S \cup \{\chi_i\} \not\models \textit{pot\_sec\_disj}$, holds too, for if it wouldn't, then via induction we also had $log_i := log_{i-1} \cup \{\chi_i\} \models \textit{pot\_sec\_disj}$, contradicting Lemma 2.
2. First we show that the first case is not triggered under db^S. From case 2 under db it follows that $log_i := log_{i-1} \cup \{\neg\chi_i\}$ and thus for db_i^S via Lemma 3 that $eval^*(\chi_i)(db_i^S) = \neg\chi_i$. For the same reasons we have $eval^*(\chi_i)(db_{i-1}^S) = \neg\chi_i$ which falsifies equation (22). Also, equation (23) is not true; otherwise we had $log_{i-1}^S \cup \{\neg\chi_i\} \models \textit{pot\_sec\_disj}$ contradicting Lemma 2 since we have that log_{i-1} under db is the same as log_{i-1}^S under db^S.
   Also our condition for case 2, namely equation (24) depends only on that log and it follows that case 2 is also triggered under db^S.
3. With the above argument it follows that case 2 is not triggered under db^S if it wasn't triggered under db. Therefore we now have to show that case 3 is triggered under db^S. From case 3 under db we have that $log_i = log_{i-1} \cup \{\neg\chi_i\} \cup \{neg(\neg \textit{con\_conj}, \chi_i)\}$. From Lemma 3 it follows that $db_i^S \ \textit{model\_of}\ log_i$ and via (28) we get $db_{i-1}^S = db_i^S$ resulting in $db_{i-1}^S \ \textit{model\_of}\ log_i$. From this follows

$$eval\big(neg(\neg \textit{con\_conj}, \chi_i)\big)\big(db_{i-1}^S\big) = \textit{true}. \quad (30)$$

   From $db_{i-1}^S \ \textit{model\_of}\ log_i$ and $\neg\chi_i \in log_i$ we have that $eval^*(\chi_i)(db_{i-1}^S) = \neg\chi_i$, enabling the usage of Lemma 1 on (30):

$$eval\big(\neg \textit{con\_conj}\big)\big((db_{i-1}^S \setminus \{\neg\chi_i\}) \cup \{\chi_i\}\big) = \textit{true} \quad (31)$$

Applying the definition of *eval* we get

$$eval\big(con\_conj\big)\big((db_{i-1}^S \setminus \{\neg\chi_i\}) \cup \{\chi_i\}\big) = false \tag{32}$$

This is equation (25), which the preceding arguments showed to hold under db^S. We also have equation (26) to be true since it only depends on the log of round $i-1$. It thus follows that case 3 is triggered under db^S, too.

4. Finally it remains to be shown that case 3 isn't triggered under db^S if db triggered case 4. From case 4 under db and Lemma 3 we have

$$db_i^S \ \ model\_of\ log_i = neg(log_{i-1}, \chi_i) \cup \{\chi_i\} \cup constraints \tag{33}$$

This gives us $eval^*(con\_conj)(db_i^S) = true$ resulting in case 3 not being triggered due to the falseness of equation (25).

7 Reversibility

A user expects to be able to undo an update he issued. This is in fact one of the two requisites Bancilhon and Spyratos require for a set of view updates to be called complete (see [BS81]). We therefore demand and prove:

Theorem 2. *For any instance of Controlled Query Evaluation with view updates as defined by Def. 4 and Def. 5 and any $\chi \in LIT$ it is true that an operation update(χ) at time $i-1$ can be successfully undone at time i by the operation update$(\neg\chi)$.*

To prove this theorem, we need the following lemmas, which we state here without a justification, due to the lack of space.

Lemma 4. *For all sets of formulas $Q, P \subseteq \mathcal{P}(FORMULA)$ and all literals $\chi \in LIT$ we have $P \models Q \Leftrightarrow neg(P,\chi) \models neg(Q,\chi)$.*

Lemma 5. *If at the point in time $i-1$ the update update(χ) has been performed successfully, i.e., case 4 of the secure view update algorithm applies, then the following property holds: $neg(log_{i-1}, \chi) \cup \{\neg\chi\} \cup constraints \not\models pot\_sec\_disj$.*

Proof (of Theorem 2). To outline the proof of the theorem, we assume that the update χ has been successfully completed at the point in time $i-1$. According to case 4 of the algorithm, we then have

$$log_{i-1} = neg(log_{i-2}, \chi) \cup \{\chi\} \cup constraints \tag{34}$$

We then have to verify that an $update(\chi_i)$ that has the form $update(\neg\chi)$ will be completed successfully, too, i.e., the cases 1 to 3 do not apply.

Case 1. We show that neither (22) nor (23) are satisfied. By (34), we have $\chi \in log_{i-1}$, so $log_{i-1} \cup \{\neg\chi\}$ is inconsistent and thus $log_{i-1} \cup \{\neg\chi\} \models pot\_sec\_disj$. This is a contradiction to (22). By Lemma 2, we have $log_{i-1} \not\models pot\_sec\_disj$, and as $\chi \in log_{i-1}$ also $log_{i-1} \cup \{\chi\} \not\models pot\_sec\_disj$, which is a contradiction to (23).

Case 2. This case cannot occur owing to Lemma 5.

Case 3. Consider $db' := (db_{i-1} \setminus \{\chi\}) \cup \{\neg\chi\}$, i.e., that instance that would become db_i, if db' satisfied the consistency constraints. First, assume $\neg\chi \in db_{i-2}$. Then $db' = db_{i-2}$, and since $eval(con\_conj)(db_{i-2}) = true$ we conclude that db', i.e., the potential db_i, will be consistent after the update $\neg\chi$.

Second, assume $\chi \in db_{i-2}$. If $eval(con\_conj)(db') = true$, the denial condition (25) does not hold. Otherwise, if $eval(con\_conj)(db') = false$, we will derive a contradiction. Under the assumptions made, we would have:

$$log_{i-1} \cup \{\chi\} \cup \{neg(\neg con\_conj, \chi)\} \not\models pot\_sec\_disj$$

Then, stepwise applying Lemma 4, the definition of *neg*, equation (34) and the idempotence of *neg*, we would finally get the result:

$$log_{i-2} \cup \{\neg\chi\} \cup neg(constraints, \chi) \cup \{\neg\chi\} \cup \{\neg con\_conj\} \not\models neg(pot\_sec\_disj, \chi)$$

This result cannot hold, since the premise is inconsistent, owing to $constraints \subseteq log_{i-2}$ on the one hand and $\neg con\_conj$ occurring on the other hand.

8 Conclusion

Data manipulation comprises queries and *updates* under preservation of *constraints*, and both kinds of operation might enable a user to infer information to be kept secret. In this paper, we extend previous insight about ensuring inference-free query answers to an original proposal for processing update requests in an *inference-free* way. The extension applies to both the formal *specification* of the confidentiality requirement and the *enforcement* mechanism.

Basically, the adapted requirement expresses that, from the user's point of view, the dynamically evolving actual instances of the information system are *indistinguishable* from alternative instances in which the protected information does not hold. Roughly summarised, for any single operation, whether a query or an update, the enforcement maintains a global *invariant*, which states that the current knowledge of the user does not imply the disjunction of the potential secrets. Under additional precautions, the local assurance suffices to guarantee the global goal of indistinguishability. Furthermore, our proposal complies to the basic rules of *acceptability* and *reversibility*, required for traditional view update mechanisms. The proposal is also in accordance with *polyinstantiation*, which is used in multilevel secure systems as an inevitable feature to resolve conflicts between preservation of constraints and hiding of confidential information.

In this paper, we have dealt with update requests issued by a user; in complementary work, we are studying updates triggered by an administrator, which leads to the problem of inference-free "view refreshments". Additionally, we are combining both kinds of updates. Interestingly, the complementary work strongly suggests to consider also transactions rather than only elementary updates.

There are many further challenging problems, including an extension to first-order logic, information systems permitting open queries, and an exploration of explicit refusals on a requested updates while avoiding the known threats of meta-inferences, as well as suitable combinations of lying and refusal. An option

to avoid distortions by lying could be worthwhile for applications where returning unreliable information is not acceptable (see [BB01, BB04a, BB04b]). Moreover, all investigations could be generalised to incomplete information systems.

References

[BS81] Bancilhon, F., Spyratos, N.: Update semantics of relational views. ACM Trans. Database Syst. 6(4), 557–575 (1981)

[BB01] Biskup, J., Bonatti, P.A.: Lying versus refusal for known potential secrets. Data Knowl. Eng. 38(2), 199–222 (2001)

[BB04a] Biskup, J., Bonatti, P.A.: Controlled query evaluation for enforcing confidentiality in complete information systems. Int. J. Inf. Sec. 3, 14–27 (2004)

[BB04b] Biskup, J., Bonatti, P.A.: Controlled query evaluation for known policies by combining lying and refusal. Ann. Math. Art. Intell. 40, 37–62 (2004)

[BB07] Biskup, J., Bonatti, P.A.: Controlled query evaluation with open queries for a decidable relational submodel. Ann. Math. Art. Intell. 50, 39–77 (2007)

[BW08] Biskup, J., Weibert, T.: Keeping secrets in incomplete databases. Int. J. Inf. Sec. 7, 199–217 (2008)

[BP06] Bohannon, A., Pierce, B.C., Vaughan, J.A.: Relational lenses: a language for updatable views. In: PODS 2006, pp. 338–347. ACM, New York (2006)

[BK95] Bonatti, P.A., Kraus, S., Subrahmanian, V.S.: Foundations of secure deductive databases. IEEE Trans. Knowledge and Data Engineering 7(3), 406–422 (1995)

[CG99] Cuppens, F., Gabillon, A.: Logical foundation of multilevel databases. Data Knowl. Eng. 29, 259–291 (1999)

[CG01] Cuppens, F., Gabillon, A.: Cover story management. Data Knowl. Eng. 37, 177–201 (2001)

[DB82] Dayal, U., Bernstein, P.A.: On correct translation of update operations on relational views. ACM Trans. Database Systems 8, 381–416 (1982)

[DA87] Denning, D.E., Akl, S., Heckman, M., Lunt, T., Morgenstern, M., Neumann, P., Schell, R.: Views for multilevel database security. IEEE Trans. Software Eng. 13(2), 129–140 (1987)

[FJ02] Farkas, C., Jajodia, S.: The inference problem: a survey. SIGKDD Explor. Newsl. 4(2), 6–11 (2002)

[He04] Hegner, S.J.: An order-based theory of updates for relational views. Ann. Math. Art. Intell. 40, 63–125 (2004)

[JS91] Jajodia, S., Sandhu, R.S.: Towards a multilevel secure relational data model. In: Proc. ACM SIGMOD Int. Conf. on Management of Data, May 1991, pp. 50–59 (1991)

[La90] Langerak, R.: View updates in relational databases with an independent scheme. ACM Trans. Database Systems 15, 40–66 (1990)

[LD90] Lunt, T.F., Denning, D.E., Schell, R.R., Heckman, M., Shockley, W.R.: The SeaView security model. IEEE Trans. Software Eng. 16(6), 593–607 (1990)

[SJ92] Sandhu, R.S., Jajodia, S.: Polyinstantiation for cover stories. In: Deswarte, Y., Quisquater, J.-J., Eizenberg, G. (eds.) ESORICS 1992. LNCS, vol. 648, pp. 307–328. Springer, Heidelberg (1992)

[SJ83] Sicherman, G.L., de Jonge, W., van de Riet, R.P.: Answering queries without revealing secrets. ACM Trans. Database Systems 8(1), 41–59 (1983)

[WS94] Winslett, M., Smith, K., Qian, X.: Formal query languages for secure relational databases. ACM Trans. Database Systems 19(4), 626–662 (1994)

Implementing Reflective Access Control in SQL

Lars E. Olson[1], Carl A. Gunter[1], William R. Cook[2], and Marianne Winslett[1]

[1] University of Illinois at Urbana-Champaign
[2] University of Texas

Abstract. *Reflective Database Access Control (RDBAC)* is a model in which a database privilege is expressed as a database query itself, rather than as a static privilege in an access control matrix. RDBAC aids the management of database access controls by improving the expressiveness of policies. The Transaction Datalog language provides a powerful syntax and semantics for expressing RDBAC policies, however there is no efficient implementation of this language for practical database systems. We demonstrate a strategy for compiling policies in Transaction Datalog into standard SQL views that enforce the policies, including overcoming significant differences in semantics between the languages in handling side-effects and evaluation order. We also report the results of evaluating the performance of these views compared to policies enforced by access control matrices. This implementation demonstrates the practical feasibility of RDBAC, and suggests a rich field of further research.

1 Introduction

Current databases use a conceptually simple model for access control: the database maintains an access control matrix (ACM) describing which users are allowed to access each database resource, along with which operations each user is allowed to use. If a user should only be granted access to certain portions of a database table, then a separate view is created to define those portions, and the user is granted access to the view. This model is flexible enough to allow users to define access privileges for their own tables, without requiring superuser privileges. However, ACMs are limited to expressing the extent of the policy, such as "Alice can view data about Alice," "Bob can view data about Bob," *etc.*, rather than the intent of the policy, such as "each employee can view their own data." This makes policy administration more tedious in the face of changing data, such as adding new users, implementing new policies, or modifying the database schema. Many databases attempt to ease administration burdens by implementing roles in addition to ACMs to group together common sets of privileges, but this does not fully address the problem. In a scenario such as the policy of "each employee can view their own data in the table," each user requires an individually-defined view of a table as well as a separate role to access each view, which yields no benefit over a standard ACM-based policy.

Reflective Database Access Control (RDBAC) is an access control model that addresses this problem [16]. We define a policy as *reflective* when it depends on data contained in other parts of the database. While most databases already

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 17–32, 2009.

do store ACMs within the database itself, the policy data are restricted to the form of a triple ⟨*user, resource, operation*⟩ and separated from the rest of the database; and the query within the policy is limited to finding the permission in the ACM. RDBAC removes these restrictions and allows policies to refer to any part of the database.

The goal of this paper is to describe how an RDBAC system can be implemented in a standard SQL-based relational database management system, using Transaction Datalog ($\mathcal{TD}$) as a policy language, which provides a theoretical formalism for expressing policies that is also compact and conceptually easy to understand. $\mathcal{TD}$ is an extension to classical Datalog that includes formal semantics for atomicity and for database updates [2]. Updates are encoded as *assertion predicates* and *retraction predicates* to insert or remove data from the database state, respectively. For example, the assertion predicate `ins.a(1,2)` (respectively, the retraction predicate `del.a(1,2)`) changes the database state to add (respectively, remove) the tuple `(1,2)` in table `a`. Using a language such as $\mathcal{TD}$ enables us to apply security to an overall transaction, rather than to each individual query in the transaction. This type of security policy is not available in transaction managers for general-purpose databases, and must be enforced at the application level. We chose to use $\mathcal{TD}$ due to its formal semantics that enable provable security properties for certain policies [16]. Similar compilation strategies applied to other more common policy languages, such as XACML [15], could be implemented.

Benchmark Policies. For the purposes of this paper we will use an example database that contains data for a consulting firm that contains branch offices in multiple locations. The database includes tables for employee data, financial records for each location, and data for the firm's clients.

The company has various access policies for the data. The user Alice creates all tables relevant to this scenario, and is allowed full access to them. All HR personnel are allowed full access to the employee data. Regional managers are allowed access to data of the employees in their region, indicated by the ID of the store in which they work: stores 100-199 are in region 1, stores 200-299 are in region 2, *etc.* The company also grants insurance agents access to employees' names and addresses, but only for those employees who have opted to release their data, and all accesses by insurance agents must be audited.

Each store location has an owner, who is allowed to view all financial records for that location. An owner may own more than one location.

Finally, the firm's clients may pose conflicts of interest. A Chinese Wall policy [4] is imposed on data for such clients: any employee may initially view any client's data, but after viewing the data, the employee may not view any other data that creates a conflict of interest.

Table 1 contains access rules for the `employees` and `store_data` tables that implement these policies, encoded in $\mathcal{TD}$. We call the predicates defined by these rules *view predicates*. Rule 1 is defined for a particular user `'alice'` and is true for all rows in the `employees` table. In other words, the view on table `employees` for user `'alice'` is the entire table.

Table 1. Benchmark Policies: `employees` and `store_data` tables

```
% base policy, automatically generated
1. view_employees('alice', Name, Addr, StoreID, Salary, Optin) :-
   employees(Name, Addr, StoreID, Salary, Optin).

% hr policy
2. view_employees(User, Name, Addr, StoreID, Salary, Optin) :-
   view_hr('alice', User),
   view_employees('alice', Name, Addr, StoreID, Salary, Optin).

% manager policy
3. view_employees(User, Name, Addr, StoreID, Salary, Optin) :-
   view_manager('alice', User, Region),
   view_employees('alice', Name, Addr, StoreID, Salary, Optin),
   >=(StoreID, Region*100), <(StoreID, (Region+1)*100).

% insurance policy
4. view_employees(User, Name, Addr, null, null, null) :-
   view_insurance('alice', User),
   view_employees('alice', Name, Addr, _, _, Optin), =(Optin, 'true'),
   ins.accesslog(User, Name, 'Name & Addr', current_time).

% policy for branch office data
5. view_store_data(User, StoreID, Data1, Data2) :-
   view_owner('alice', StoreID, User),
   view_store_data('alice', StoreID, Data1, Data2).
```

Rules 2 and 3 demonstrate how we can leverage the recursive semantics of Datalog to define other useful policies. In Rule 2, the first predicate in the body of the policy is true if and only if the querying user is in the `hr` table. The second predicate, as we explained for Rule 1, is true for all records in the `employees` table. In other words, this rule enforces the policy that any HR user can see the data for all employees. In Rule 3, the first predicate in the body of the policy is true if and only if the querying user is in the `manager` table; if so, the variable `Region` is bound to the value of the region that manager is assigned to. The second predicate is true for all employees in the table, however the final two predicates are only true for the employees with a StoreID number between `Region` * 100 and (`Region` + 1) * 100. In other words, managers can see the data for all employees in their region.

Rule 4 uses one of the extensions defined in $\mathcal{TD}$ to implement an audit policy. The first predicate in the body checks whether the querying user is an authorized insurance agent. The second retrieves the names and addresses of each employee, but uses "don't care" values for the StoreID, Salary, and Optin fields (represented by the underscore character). The corresponding fields in the head of the rule contain `null` values. The third predicate filters the table to only those rows

Table 2. Benchmark Policies: client data

```
% Chinese Wall policy
6. view_client1(User, Data1, Data2) :-
   view_cwUsers('alice', User, 1, OldVal),
   view_client1('alice', Data1, Data2),
   del.cwUsers(User, 1, OldVal), ins.cwUsers(User, 1, 0).

7. view_clientData(User, Client, Data1, Data2) :-
   view_cwUsers('alice', User, 1, OldVal),
   view_clientData('alice', Client, Data1, Data2), =(Client, 'client1'),
   del.cwUsers(User, 1, OldVal), ins.cwUsers(User, 1, 0).
```

with the `Optin` value set to `true`. This demonstrates how cell-level security, using both column-level and row-level restrictions, can be implemented with a $\mathcal{TD}$ rule. Finally, an audit record containing the name of the querying user, the name of the user whose record is accessed, an explanatory string, and the current time is added to the `accesslog` table for each user accessed.

Rule 5 implements the policy for the `store_data` table that branch office owners can view data for the offices they own. This rule depends on data from other policies for the `store_data` table and an `owner` table, which are omitted for brevity.

Table 2 contains two alternative rules for implementing the Chinese Wall policy that protects client data, depending on whether each client's data is stored in a separate table (Rule 6), or a single table contains the data for all clients (Rule 7). We provide rule 7 only to demonstrate the expressive power of RDBAC; hereafter we will only use rule 6. Either alternative requires some initial setup in creating a table called `cwUsers` with the following schema: `User` of type `varchar`, `CanAccessClient1` of type `int`, and `CanAccessClient2` of type `int`. The last two columns could equivalently be defined as booleans. Initially, this table contains a row for every authorized employee, with both columns set to 1. The first predicate in rule 6 checks whether the user is allowed to access the `client1` table, based on whether his entry in the `cwUsers` table has a 1 in the `CanAccessClient1` column. Assuming this is satisfied, the second predicate returns the data requested by the user. The third and fourth predicates remove the user's row from the `cwUsers` table, whatever the value of `CanAccessClient2` may have been, and replace it with an entry that only allows future access to the `client1` data and turns off future access to the `client2` data. Other rules, not shown, would be similarly defined for accessing the data for `client2`, except reversing the columns on the `cwUsers` table: they would check whether `CanAccessClient2` is 1, and would set `CanAccessClient1` to 0.

The access control model used by most modern commercial databases offers a compelling case for decentralized policy administration, in which table and view owners define their own access control policies for the resources they create. Ideally, more advanced access control models should still give users the same kind of autonomy in defining their own policies. However, this autonomy comes

at a price. We have shown that careless definitions of reflective policies can be vulnerable to a Trojan Horse attack if untrusted users are also allowed to define policies [16]. This problem can easily be mitigated using $\mathcal{TD}$-based policies by restricting a policy definer from performing any operations beyond what that user can perform manually: we simply make the user's ID an explicit parameter to all view predicates, and for all predicates in the body of a policy, that parameter is bound to the ID of the policy definer, thereby executing the policy under the policy definer's privileges. The database system can automatically generate basic privileges that access the table directly, such as the first rule in Table 1, to the table owner. All other user-defined policies must query the database through the view predicates.

The rest of the paper is divided into three sections. In Section 2 we describe a prototype implementation of our RDBAC system, including challenges in matching $\mathcal{TD}$ semantics with SQL. In Section 3 we evaluate the performance of our system, compared to a baseline implementation that uses static ACM-based policies. We describe related work and conclude in Section 4.

2 Implementation

2.1 Strategy

Our goals in implementing a prototype system to demonstrate the usability of a reflective database access control system included the following: use a flexible, expressive policy definition language; use, as much as possible, an existing database management system following the SQL standard; minimize the overhead running time for executing queries; and allow scalability both in the number of users and in the amount of data stored.

$\mathcal{TD}$ provides a very concise syntax that is capable of expressing a wide range of policies, as demonstrated in Section 1. Translating classical Datalog rules into SQL statements has been well-studied in the past [10, 11] and we took a similar approach for our prototype, in which we compile a set of $\mathcal{TD}$ rules into a set of SQL view definitions. These view definitions can then be added to the database and used normally, with no additional overhead. Because rules may be recursively defined, it was necessary to use a database system that implements recursive views as defined by the SQL:1999 standard. We chose to use Microsoft's SQL Server 2005.

Unfortunately, there are two significant semantic gaps between $\mathcal{TD}$ and SQL. One problem is that SQL does not allow database update statements within a data retrieval query. SQL triggers, while designed to perform updates as side-effects to user actions, cannot be defined for read-only queries. In some databases, the restriction against side-effects can be bypassed by calling a user-defined function (UDF) from within the query which performs the update. Other databases, including SQL Server, preclude this by disallowing the invocation of any function that causes side-effects on the database from within read-only queries. Indeed, this is generally a safer configuration; otherwise, functions with side-effects could be invoked without the user's knowledge, causing a vulnerability with

Trojan-Horse code. However, we have argued that under certain reasonable conditions, such code can be analyzed to prevent undesirable side-effects [16]. One workaround for executing side-effects in SQL Server is to execute it from within a Common Language Runtime (CLR) function, which can then be registered as an external function within the SQL Server database. This workaround is not an ideal solution; it is considered an egregious hack [14] that requires a separate connection to the database, which adversely affects performance. However, it suffices for a proof-of-concept prototype.

The other semantic gap between $\mathcal{TD}$ and SQL is that the former includes a well-defined execution ordering in its definition, the latter does not. In other words, SQL provides no way to distinguish the policies a_1 `:-` b`,` c and a_2 `:-` c`,` b. For traditional SQL queries, this is advantageous to the query optimizer, which can reorder query plans to find highly efficient executions of the query. However, there are two reasons why lack of ordering is a cause for concern in implementing $\mathcal{TD}$: the compiled SQL view may not be a valid execution of the original $\mathcal{TD}$ rule, and the order of operations in a query plan may cause information leakage [12]. To solve both problems, our prototype only implements policies in which all side-effects occur at the end of the policy execution, after all relevant data has been retrieved and filtered. It combines all of the read operations into a subquery along with dynamically-generated parameters to the UDF that executes the side-effect, making the side-effects dependent on the results of the read operations and thus ensuring that the execution order is followed and preventing information leakage by guaranteeing that no side-effects will occur until it knows the transaction will definitely run to completion.

Our approach for implementing RDBAC policies is to write a compiler that parses a $\mathcal{TD}$-based policy and generates a standard SQL:1999 view definition that enforces the policy. In order to demonstrate the compilation process, we will walk through an example of the process using rule 4 of our benchmark policies from Table 1.

On the first pass, our compiler determines the schema for the view, comprised of an explicit parameter for the user executing the query, the schema of the base table, and parameters for any assertions or retractions that any rule for that view might execute. In this case, the generated schema is: `User`, `Name`, `Addr`, `StoreID`, `Salary`, `Optin`, `Assert_flag` (a boolean flag to indicate whether the rule triggers the assertion), `Assert_param0`, and `Assert_param1` (parameters to the UDF that will execute the assertion).

The compiler also generates a UDF that creates and executes an SQL `insert` statement, corresponding to the assertion predicate `ins.accesslog(User, Name, 'Name & Addr', current_time)`. The values for the string constant `'Name & Addr'` and the keyword `current_time` can be directly translated into the generated `insert` statement (the latter is translated to the built-in function `GETDATE()`). The other parameters, `User` and `Name` (not to be confused with the schema attributes `User` and `Name`), are determined at execution time.

During the second pass, the compiler maintains a list of tables and views accessed in the rule (which will form the SQL `FROM` clause), a list of variables

and variable bindings that appear in the rule, and a list of conditions imposed by built-in predicates or constants (which together will form the SQL `WHERE` clause). After this information is gathered, it uses the variable bindings to form the list of attributes to appear in the view (which will form the SQL `SELECT` clause).

First, it examines the body of the rule. The first literal is the view predicate `view_insurance('alice', User)`. The view `view_insurance` is added to the `FROM` clause list, and given an alias `i`. The constant `'alice'` adds a condition to the `WHERE` clause; using the available metadata for the view, the compiler determines that this condition should be `i.User = 'alice'`. The predicate variable `User` (not to be confused with the table attribute `i.User`) is bound to the second attribute in `view_insurance`, `i.Name`, which is added to the list of variable bindings.

Similarly, the second literal is the view predicate `view_employees('alice', Name, Addr, _, _, Optin)`. The view `view_employees` is added to the `FROM` clause list and given an alias `e`. The constant `'alice'`, together with the metadata for the view, indicates that `e.User = 'alice'` is added to the `WHERE` clause list. The variables `Name`, `Addr`, and `Optin` are bound to the attributes `e.Name`, `e.Addr` and `e.Optin`, respectively, all of which are added to the list of variable bindings. The don't-care symbols (underscores) are ignored, since they impose no conditions on the values in the view.

The third literal is the built-in predicate `=(Optin, 'true')`. Because the variable `Optin` was bound to the attribute `e.Optin`, this adds the condition `e.Optin = 'true'` to the `WHERE` clause list.

Next, the fourth literal is the assertion predicate `ins.accesslog(User, Name, 'Name & Addr', current_time)`. As previously described, during the first pass this literal triggered the creation of a UDF. During the second pass, the compiler uses the list of variable bindings to determine which values will be passed as parameters to this function, contained in the view schema as `Assert_param0` and `Assert_param1`. In this case, `i.Name` is added to the `SELECT` clause list as the former, and `e.Name` is added as the latter. The value for `Assert_flag` is also added to the `SELECT` clause list as 1, indicating that the side-effect should be executed. For the other rules, which do not contain assertion predicates, the value for `Assert_flag` is added as 0, and the other parameters are given null values.

Finally, the head literal is examined to determine the attributes that should appear in the `SELECT` clause. The `User` variable, bound previously to `i.Name`, is added as `User`. Similarly, `e.Name` is added as `Name` and `e.Addr` is added as `Addr`. The constant `null` is added as the other selected attributes: `StoreID`, `Salary`, and `Optin`. In order for the recursive view to compile properly, SQL Server requires that the `null` values be cast with the proper types, which can be retrieved from the metadata for the `view_employees` view.

The other rules are similarly translated, and connected by the `UNION ALL` operator. Finally, the compiler takes the result of the union and passes the appropriate parameters into each UDF. The complete[1] generated view for all the policies for `view_employees` is shown in Table 3, along with another automatically-generated

[1] For clarity, the cast operations required by SQL Server for its recursive query definitions have been omitted from the view presented here.

Table 3. Generated SQL view definition for benchmark policies

```
--...function assert_accesslog definition omitted...
create view view_employees as
  with view_employees as (
    select 'alice' as User, e.Name as Name, e.Addr as Addr, e.StoreID as
      StoreID, e.Salary as Salary, e.Optin as Optin, 0 as Assert_flag,
      NULL as Assert_param0, NULL as Assert_param1
    from employees e                                    -- base policy
  union all
    select h.Name as User, e.Name as Name, e.Addr as Addr, e.StoreID as
      StoreID, e.Salary as Salary, e.Optin as Optin, 0 as Assert_flag,
      NULL as Assert_param0, NULL as Assert_param1
    from view_employees e, view_hr h
    where e.User = 'alice' and h.User = 'alice'      -- hr policy
  union all
    select m.Name as User, e.Name as Name, e.Addr as Addr, e.StoreID as
      StoreID, e.Salary as Salary, e.Optin as Optin, 0 as Assert_flag,
      NULL as Assert_param0, NULL as Assert_param1
    from view_employees e, view_manager m where e.User = 'alice'
    and m.User = 'alice' and e.StoreID >= m.Region*100
    and e.StoreID < (m.Region+1) * 100               -- manager policy
  union all
    select i.Name as User, e.Name as Name, e.Addr as Addr, NULL as
      StoreID, NULL as Salary, NULL as Optin,
      1 as Assert_flag, i.Name as Assert_param0, e.Name as Assert_param1
    from view_employees e, view_insurance i where e.User = 'alice'
    and i.User = 'alice' and e.Optin = 'true'        -- insurance policy
  ) select distinct User, Name, Addr, StoreID, Salary, Optin
  from view_employees where (Assert_flag = 1 and assert_accesslog(
    Assert_flag, Assert_param0, Assert_param1) != 0) or Assert_flag = 0;
create view view_employees_public as
  select Name, Addr, StoreID, Salary, Optin from view_employees
  where User = CURRENT_USER;
grant select on view_employees_public to public;
```

view `view_employees_public` that queries `view_employees` on behalf of the current user and may be safely queried by any user in the system. The portion of the generated code which we stepped through is indicated by the comment "-- *insurance policy*."

2.2 Optimization

Translating the view definition into standard SQL allows the execution of reflective access policies to take advantage of the large body of work in query optimization that has been implemented in commercial databases. There are also additional possible optimizations we developed using partial evaluation techniques [3] on the $\mathcal{TD}$ rules.

As described in Section 1, our system prevents information leakage in reflective policies by forcing them to run under the definer's privileges. Only the basic privileges, defined automatically by the database management system, access the tables directly. Thus, all user-defined privileges are by nature recursive, since they must in turn be based on another access rule. However, it should be noted that without this restriction, we can sometimes define equivalent policies that are recursion-free. For example, the second, third, and fourth rules in Table 1 all depend on the first rule. Since we know from the first rule that the user `'alice'` is allowed access to the entire `employees` table with no restrictions or filters, the compiler could have simply replaced the references with direct accesses to the table.

This suggests that unfolding predicates in the rule before compiling it to an SQL view could yield a significant performance benefit. While more complex partial evaluation techniques would require a sophisticated $\mathcal{TD}$ interpreter, it is simple to keep track of the basic privileges and unfold them into the rules in which they appear. In our running example of the policies from Table 1, using this optimization generates code that is similar to the generated view in Table 3; hence, we have not included it. The key difference is that each subselect accesses the tables `employees`, `hr`, *etc.* directly, rather than through the views `view_employees`, `view_hr`, *etc.*

Removing the recursion from a view also enables us to remove the redundant `cast` operations, as SQL Server is better able to match types in recursion-free views. Additionally, the `select distinct` to remove duplicate rows at the end of the union is another costly operation. While this operation technically ensures that the result strictly adheres to the semantics of $\mathcal{TD}$, removing it still yields the same answer set in our test cases, and it would similarly be redundant in many other practical cases. We have implemented all of these optimizations in our prototype system, which we assess in Section 3.

An opportunity for further optimization would be to pre-compute the parts of the view that are checks on the user's identity. For instance, consider the policy rules from Table 1. If a given user is recorded in the `insurance` table but not in any other table, when that user logs in, the database could partially compute the view to determine that only rule 4 is applicable to this user, not rules 1, 2, or 3. This would enable us to avoid calculating extraneous `UNION ALL` operations by constructing the view definition dynamically. We have written a simulated version of such an optimization using a stored procedure, and assess its performance in Section 3 as well.

3 Evaluation

We evaluated the execution time of running queries on views generated from the benchmark policies by our implementation, such as the view from Table 3, to a baseline of running queries on custom-written views, such as those in Table 4. To test these views, we used Microsoft's SQL Server 2005 database management system, running on a 2.4 GHz Intel Core2 machine with Windows Vista Business

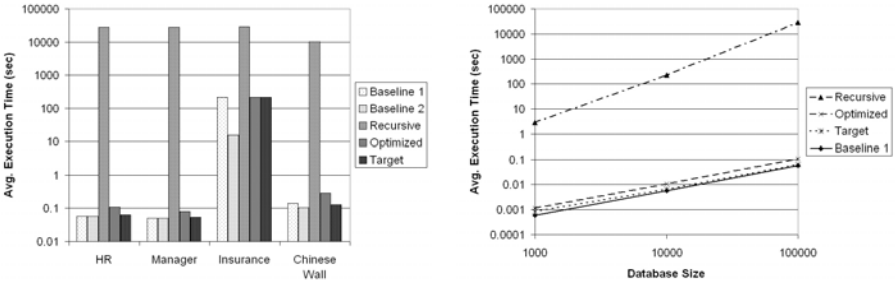
Table 4. Hand-coded baseline SQL views

```
-- base policy
grant select on employees to alice;

-- hr policy
grant select on employees to {username(s)};

-- manager policy
create view region1_view as
  select * from employees where StoreID >= 100 and StoreID < 200;
grant select on region1_view to {username(s)};
create view region2_view as
  select * from employees where StoreID >= 200 and StoreID < 300;
grant select on region2_view to {username(s)};
-- similar views created for each region

-- insurance policy
create view insurance_view as
  select Name, Addr from employees where Optin='true' and
  assert_accesslog(CURRENT_USER, Name, 'Name & Addr', GETDATE())=true;
grant select on insurance_view to {username(s)};
```

64-bit Edition. The base tables all have appropriately-defined indexes on the user names, in order to minimize the cost of performing joins.

Each test was performed using an external application written in C# and compiled by Microsoft's Visual C# 2008 compiler version 3.5. The application was run locally so as not to include network latency. For each user, the application constructed a query for the entire table and iterated through each row of the table. The query was repeated until the query time reached a stable state, after which we gathered multiple execution times, of which we report the average query time. Thus, our results represent the time for "hot" queries, or queries which have been loaded and executed recently.

We tested two versions of our prototype code: one which directly translates the policies into a recursive view, and another which performs the unfolding optimization defined in Section 2.2. We also tested a simulated version of the partial-evaluation optimization, also described in Section 2.2, using a stored procedure. To assess the scalability of the generated views, the experiment was repeated on databases with 1000 users, 10,000 users, and 100,000 users, each with a record in the `employees` table. The size of the `hr`, `manager`, and `insurance` tables also increase proportionally in each experiment, with 100 entries each, 1000 entries each, and 10,000 entries each, respectively. For brevity, we will not include the raw results here, but will make them available in a later publication. Figure 1(a) shows these results graphically for the database with 100,000 users, and Figure 1(b) shows the results of querying each view as an HR user as the database size increases from 1000 to 100,000. Queries from the other users scaled similarly,



(a) Fixed database size (100,000 empl.) (b) Fixed query type (HR query)

Fig. 1. Comparison of execution time results (in sec) for `employee` policies

so those results are not shown. The data labeled "Baseline 1" and "Baseline 2" are the results of querying the hand-coded views from Table 4. The difference between the two baselines lies in how they handle queries that contain side-effects. For the first baseline, the view directly calls a UDF that executes the side-effect. For the second baseline, the side-effect is not enforced by the view at all, but rather by the querying application. This allows us to measure the cost of using UDFs, apart from the cost of using a compiled view. The data labeled "Recursive" are the results of querying the compiled view from Table 3, "Optimized" are the results of querying the compiled view using our predicate unfolding optimization, and "Target" are the results of executing the stored procedure that uses partial evaluation with dynamic view construction. Because Baseline 2 is no different than Baseline 1 for the HR query, we omitted the data for Baseline 2 in Figure 1(b) for clarity. Notice that on both charts we use a logarithmic scale for the execution time; in Figure 1(a) this helps demonstrate the successive improvements each optimization makes, and in Figure 1(b), this shows the scalability of executing the views as the size of the database increases exponentially.

The queries with side-effects show the cost of using the workaround in SQL Server which opens a separate connection to execute the update. Our results show that this does indeed noticeably affect all three views that use the workaround. Databases that could handle allowing side-effects in selection queries would not experience this effect as dramatically.

For the Chinese Wall query, which uses the workaround twice on each row, the recursive view behaves as expected. However, neither the optimized view nor the first baseline, both of which also use the workaround, show much effect from its use. After tracing the execution of the query, we discovered the cause of this unexpected result. SQL Server recognized that the same parameters are being passed to the UDF, and rather than re-executing the function on each row, it cached the return value after executing on the first row and used the cached value on each subsequent row. Effectively, the side-effects are being executed once per query rather than once per row. While this still correctly enforces the Chinese Wall policy (access to the other table is prohibited, whether the user queries one

row or all of the rows), the execution order is not semantically equivalent to the recursive query.

Increasing the database size shows that while the recursive view does not scale very well, the optimized view and the stored procedure handle larger data sizes much better, remaining at roughly the same proportion to the performance of the baseline views for all our tests. The workaround for executing side-effects drastically affects queries for small data sizes, so seeing the same effect on queries for large data sizes is no surprise. This should be a major focus for improvement in the future.

The unfolding optimization from Section 2.2 is clearly beneficial, since it removes the recursion from the rules, eliminating the need for executing a fixed-point algorithm to evaluate queries. The additional optimization using partial evaluation further improves the performance to nearly as fast as the baseline.

In cases where fewer policies protect a table, the performance of the compiled view is even better. Recall the policy for the `store_data` table from Table 1. This policy poses additional administration difficulties when using traditional ACM-based approaches, which are automatically solved by an RDBAC-based approach. Because a single owner may need access to multiple parts of the table, and there is no simple, single encapsulation of the conditions describing all of these parts, a traditional ACM-based view requires more complicated conditions than those described in the baseline for the `employees` table. These more complicated conditions therefore become more difficult to keep updated when the data or the permissions on the data change.

We describe four baselines for querying this table using traditional ACM-based views, each of which requires a somewhat different configuration by the database administrator. One approach, which we will call "Union Baseline," is where the administrator creates a separate view for each franchise, and the owner executes a `union all` over each view. A similar approach, "App-Level Baseline," queries each view individually but aggregates the data at the application level, rather than at the database level. These two approaches minimize the work needed when a store location is sold to a different owner, requiring only one `revoke` and one `grant` statement and no view redefinitions; however, the processing times for queries using these baselines are considerably more costly, as demonstrated by the results. Another approach, "Disjunction Baseline," requires the administrator to create a customized view for each owner that includes data from each store he owns, implemented as a disjunction of the Store ID's in the `where` clause of the view. This approach makes the store owners' jobs much easier, since it does not require them to query multiple views, and also executes faster than the other two baselines. However, it also requires more upkeep by the administrator, who must redefine two views each time a location changes hands. A fourth approach, "Join Baseline," joins the data from the `owner` table, but is otherwise similar to the Disjunction Baseline in creating a customized view for each owner. This requires less upkeep from the Disjunction Baseline when owners are changed, but still requires new views to be defined when new owners are added. Note that this is nearly the same approach as described in the $\mathcal{TD}$ rule, except that the

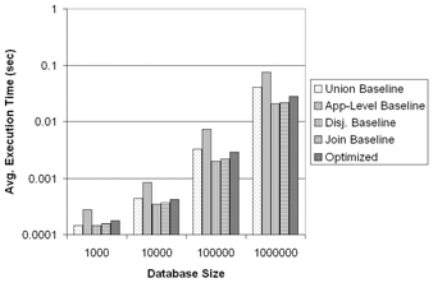


Fig. 2. Comparison of execution time results (in sec) for `store_data` policy

Join Baseline does not use a single all-purpose view that depends on the user's identity.

Figure 2 shows a graph of the results of running the optimized compiled view for the `store_data` table compared with the results of using each of the four baselines, again using a logarithmic scale. The reflective view generated by our compiler offers performance comparable to the fastest of these baselines, and requires less maintenance than any of the baselines when the data changes. Only a single view is created, and if a store location changes owners, this information simply needs to be updated in the `owner` table, which must still be maintained even when using the other baselines anyway.

4 Related Work and Conclusion

4.1 Predicated Grants

While our translation algorithm can already be used for current SQL database systems, the translation process could be made easier using a proposed extension to SQL called *predicated grants* [8], in which access rules to tables and views are defined in the SQL `grant` statement, rather than in a view definition. The `grant` syntax is extended to include a `where` clause (similar to what might be found in a `select` statement) and optionally a query-defined user group, in which the grantees are defined by the result set of another query.

For example, the *hr* policy from Table 1 can be expressed in a predicated grant as

```
grant select on employees where userId() in (select Name from hr) to public
```

or by defining a query-defined user group for all *hr* users as

```
create group hrGrp as (select Name from hr);

grant select on employees to hrGrp
```

Column-level privileges simply follow the SQL standard of listing the allowed columns after the table name, such as `grant select on employees(Name, Addr) e where e.Optin='true' to insuranceGrp`.

Predicated grants do not currently support side-effects, required by policies such as the *insurance* policy from Table 1 or the *Chinese Wall* policy from Table 2, so further extensions would be necessary to implement them. One possibility might be simply to use UDFs, as our implementation does. Another possibility might be to allow compound statements in the predicate of the grant, such as:

```
grant select on employees(Name, Addr) e where e.Optin='true' and
  userId() in (select Name from insurance i;
  insert into accesslog values(i.Name, e.Name, 'Name & Addr', GETDATE()))
  to public
```

Such an extension would facilitate a more direct translation from $\mathcal{TD}$ semantics into SQL, including execution ordering.

Besides not implementing side-effects, predicated grants also currently disallow policies that refer back to the same table they protect, as well as policy cycles. Our prototype easily handles such policies, which can occur very naturally in practice. For instance, consider the policy "all employees may view names and addresses of other employees that work in the same store." This policy protects the `employees` table, but also needs to query that table itself to find out what store the querying user works in. The authors briefly mention a prototype implementing portions of their extended syntax, however no details are provided so it is unknown how well the prototype performs.

4.2 Other Related Work

We have previously described the concept of RDBAC using $\mathcal{TD}$ in other work [16], which contains a more complete list of references for other literature describing similar concepts. In particular, Oracle's VPD technology [17] allows UDFs, possibly containing other queries itself, to be used as filters on any table or view. While this is an already-widely-deployed DBMS with reflective capabilities, this functionality is intended only for users with superuser privileges, as unskilled definers may write unsafe policies [16]. Additionally, policies that refer back to the same table they protect are also disallowed. The patent by Cook and Gannholm [9] describes another reflective system in which policies contain queries. Their system also allows policies on entire transactions as well as on individual queries, but assumes an omniscient policy definer that can access the entire database, rather than requiring such queries to satisfy policies themselves.

The relationship between the expressive powers of Datalog and relational algebra has long been recognized [1, 6, 18], although few systems that analyze the practical use of Datalog or Prolog together with database management systems have actually been built [7, 10, 11, 13]. Draxler's work offers the most details and is most similar to ours, in describing a translation process from a subset of general-purpose Prolog syntax into SQL [10]. He also offers a survey of other earlier literature describing the translation process. Disjunctions, negations, and aggregates are all supported, as are some non-Datalog features of Prolog such as `findall` and nested predicates; however it does not handle recursive view

definitions, or even views that depend on the results of other queries defined in the program (unless, of course, the query is copied verbatim, making the system susceptible to update anomalies). Additionally, applications using this interface must also be written in Prolog. The report mentions two proposed approaches to incorporating database updates in their system; however both approaches are only described at a high level, and neither appears to have been implemented. Other publicly-available translation engines from Prolog to SQL exist, but all are derived from Draxler's code base.

U-Datalog [5] is an alternative extension to Datalog that defines update semantics, in which all updates are deferred until the end of a query evaluation, similarly to the restriction on $\mathcal{TD}$ that we used in our implementation. Conflicting updates, in which reordering the updates results in a different final database state, are detected and aborted. U-Datalog could offer a reasonable alternative language to $\mathcal{TD}$; however, ordering of updates are very important in certain policies. Consider, for instance, a policy in which only one user may access a data item at one time, which could be implemented as `token(X), del.token(X), read_data, ins.token(X)`. Clearly the ordering of these predicates is significant, as other orderings may cause a policy violation or deadlock.

4.3 Conclusion and Future Work

We have described an implementation of reflective database access control based on the semantics of Transaction Datalog. This implementation compiles a set of policies into standard SQL views that can be used in current database management systems. We have evaluated this implementation and demonstrated an optimization that eliminates recursion in many common cases.

Further improvements can still be made with this work, including generalizing our algorithm further to handle view predicates on assertions or retractions; creating an enforcement mechanism that disallows unsafe policies; and augmenting $\mathcal{TD}$ with syntax for atomic update policies that may depend on both the old and the new database states, as opposed to separate policies for insertion and deletion. RDBAC theory itself is a rich field for future research, including finding other useful classes of analyzable policy configurations, and developing efficient algorithms for policy analysis.

Acknowledgements. This work was supported in part by NSF CNS 07-16626, NSF CNS 07-16421, NSF CNS 05-24695, ONR N00014-08-1-0248, NSF CNS 05-24516, NSF CNS 05-24695, DHS 2006-CS-001-000001, and grants from the MacAruthur Foundation and Boeing Corporation. Additionally, Dr. Cook is supported by NSF CCF-0448128. The views expressed are those of the authors only.

References

[1] Abiteboul, S., Hull, R.: Data functions, datalog and negation (extended abstract). In: SIGMOD Conference, pp. 143–153 (1988)

[2] Bonner, A.J.: Transaction datalog: A compositional language for transaction programming. In: Cluet, S., Hull, R. (eds.) DBPL 1997. LNCS, vol. 1369, pp. 373–395. Springer, Heidelberg (1998)

[3] Bossi, A., Cocco, N., Dulli, S.: A method for specializing logic programs. ACM Transactions on Programming Languages and Systems 12(2), 253–302 (1990)
[4] Brewer, D.F.C., Nash, M.J.: The chinese wall security policy. In: IEEE Symposium on Security and Privacy, Oakland, CA, May 1989, pp. 206–214 (1989)
[5] Catania, B., Bertino, E.: Static analysis of logical languages with deferred update semantics. IEEE Transactions on Knowledge and Data Engineering 15(2), 386–404 (2003)
[6] Ceri, S., Gottlob, G., Lavazza, L.: Translation and optimization of logic queries: The algebraic approach. VLDB, 395–402 (1986)
[7] Ceri, S., Gottlob, G., Wiederhold, G.: Efficient database access from prolog. IEEE Trans. Software Eng. 15(2), 153–164 (1989)
[8] Chaudhuri, S., Dutta, T., Sudarshan, S.: Fine grained authorization through predicated grants. In: ICDE, Istanbul, Turkey, April 2007, pp. 1174–1183 (2007)
[9] Cook, W.R., Gannholm, M.R.: Rule based database security system and method. United States Patent 6, 820, 082 (November 2004)
[10] Draxler, C.: Accessing Relational and Higher Databases Through Database Set Predicates in Logic Programming Languages. PhD thesis, Zürich University (1991)
[11] Hajiyev, E., Verbaere, M., de Moor, O.: Codequest: Scalable source code queries with datalog. In: Thomas, D. (ed.) ECOOP 2006. LNCS, vol. 4067, pp. 2–27. Springer, Heidelberg (2006)
[12] Kabra, G., Ramamurthy, R., Sudarshan, S.: Redundancy and information leakage in fine-grained access control. In: SIGMOD Conference, Chicago, IL, June 2006, pp. 133–144 (2006)
[13] Maier, D.: Is prolog a database language? In: NYU Symposium on New Directions for Database Systems, New York City (May 1984)
[14] Microsoft TechNet Forums. SQL/CLR DML error: Invalid use of side-effecting or time-dependent operator. World Wide Web electronic publication (April 2008), http://forums.microsoft.com/TechNet/ShowPost.aspx?PostID=3203413&SiteID=17
[15] OASIS. eXtensible Access Control Markup Language (XACML). Technical Report 1.1, OASIS (August 2003)
[16] Olson, L.E., Gunter, C.A., Madhusudan, P.: A formal framework for reflective database access control policies. In: CCS 2008, Alexandria, VA (October 2008)
[17] Oracle Corporation. Oracle Virtual Private Database. Technical report, Oracle Corporation (June 2005), http://www.oracle.com/technology/deploy/security/db_security/virtualprivatedatabase/index.html
[18] Ullman, J.D.: Principles of Database and Knowledge-Base Systems, vol. I. Computer Science Press (1988)

An Approach to Security Policy Configuration Using Semantic Threat Graphs

Simon N. Foley and William M. Fitzgerald

Cork Constraint Computation Centre,
Computer Science Department, University College Cork, Ireland
s.foley@cs.ucc.ie, info@williamfitzgerald.net

Abstract. Managing the configuration of heterogeneous enterprise security mechanisms is a wholly complex task. The effectiveness of a configuration may be constrained by poor understanding and/or management of the overall security policy requirements, which may, in turn, unnecessarily expose the enterprise to known threats. This paper proposes a threat management approach, whereby knowledge about the effectiveness of mitigating countermeasures is used to guide the autonomic configuration of security mechanisms. This knowledge is modeled in terms of *Semantic Threat Graphs*, a variation of the traditional Threat/Attack Tree, extended in order to relate semantic information about security configuration with threats, vulnerabilities and countermeasures. An ontology-based approach to representing and reasoning over this knowledge is taken. A case study on Network Access Controls demonstrates how threats can be analyzed and how automated configuration recommendations can be made based on catalogues of best-practice countermeasures.

1 Introduction

A significant challenge in the process of securing complex systems is attaining a degree of confidence that a security configuration adequately addresses the (security) threats. *Threat Trees* [1,2], *Attack Trees* [3] and similar tree-based threat-modeling methodologies [4,5] are used to help identify, represent and analyze about threats to an enterprise's assets. Their top-down approach provides a semi-formal and methodical way to determine viable threat vectors (who, why and how a system can be compromised). In practice these trees are used for threat elicitation and analysis: representing threats at a high-level of abstraction and they tend not to be used to capture low-level or concrete security configuration detail. For example, while a threat tree may identify a firewall countermeasure for a Denial of Service attack, it is not advantageous/intended to model, for example, the distinctions between SYN-proxy versus SYN-threshold configurations [6] for a firewall in a sub-net that is downstream from other similarly configured firewalls. In the latter case much of semantics of the threats and countermeasures must be modeled implicitly and outside of the tree structure. Threat trees are useful for analyzing threats in a local context that is decomposed from some root

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 33–48, 2009.

threat, however their advantages are diminished when one considers the threat in a global context (across multiple root threats).

In this paper we consider how a threat tree style approach can provide a basis for automatically testing whether a security configuration adequately mitigates the identified threats. In order to achieve this, we extend the threat tree model to include semantic knowledge about the security configuration and how it relates to assets, threats, vulnerabilities and countermeasures. Knowledge, such as security and network configuration, and relationships with vulnerabilities and threats, is represented using ontologies [7], providing a framework in which to extend threat trees to *Semantic Threat Graphs* (*STG*s).

Semantic threat graphs are used to model knowledge about threat mitigation by security configurations. We take the Open World Assumption [8], with the result that semantic threat graphs are easily extended to incorporate knowledge about the configuration of new threats and/or additional security mechanisms. For example, building a semantic threat graph using existing firewall and proxy ontologies [9,10] in order to describe specific syn-proxy and syn-threshold countermeasure configurations.

We can use this model to build a knowledge-base of best-practice defenses and systems security policies against known threats. For example, bogon firewall rules [11,12,13] are best-practice protection against spoofing-threats for internal servers and end-user workstations, while NIST recommend multiple countermeasures over an n-tier network hosting a Web-server [14]. This knowledge-base is searchable—a suitable countermeasure/policy can be found for a given threat—and provides the basis for autonomic security configuration.

This paper is outlined as follows. Section 2 provides an introduction to threat trees highlighting their limitation in the context of low-level configuration. Section 3 proposes semantic threat graphs as a more a natural approach to construct and analyse security policies. A formal specification of a semantic threat graph, grounded in the *NAC* domain, is modelled in Section 4. Section 5 provides a case study that describes the basis for automated analysis and synthesis of suitable catalogue configuration recommendations.

2 Threat Trees

A *threat* can be defined as *"a potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm"*[15]. Threat trees (similarly, *attack trees* [3]) provide a semi-formal way of structuring the various threats that an asset may encounter. Various extensions of the threat tree paradigm have been developed. For example, the inclusion of countermeasures within the tree provides a new kind of tree called a *Defense Tree* or *Protection Tree* [4,5]. For simplicity, and if no ambiguity arises, we refer to these approaches collectively as threat trees.

A threat tree is composed of a single root node that defines the primary threat to an asset ('Disrupt Web Server', Figure 1(a)). A threat may be decomposed into additional fine-grained sub-threats ('Denial of Service'), thereby forming

a tree hierarchy [1]. A *threat profile* can be described as the path from a leaf node to the root node which represents a specific set of states involved in either achieving the primary threat or countering it.

Everything is a Threat. Each node is either a threat or a countermeasure. In practice, threats are not viewed in isolation and additional concepts must be implicitly encoded within the nodes of the tree. For example in Figure 1(a), various enterprise *Assets* are referenced: a Web server and firewall are implicit by the 'Disrupt Web Server', 'Firewall-1' nodes respectively. Similarly the Web server's TCP/IP stack indicates an implicit *Vulnerability* in the 'Exploit 3-way Handshake' threat node. By viewing everything as a threat, implicit information may be overlooked.

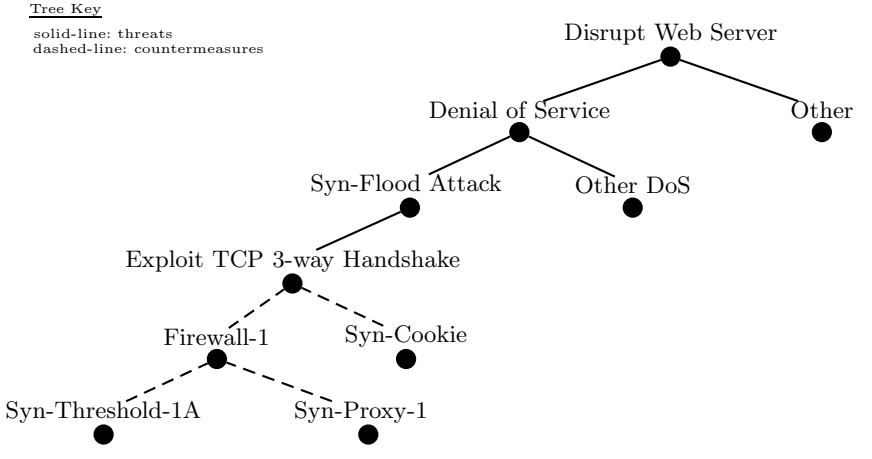
Implicit Threat Relations. A threat tree represents threat-decomposition and does not explicitly model other relationships between threats (or concepts related to threats). For example, in Figure 1(a), the 'Syn-Flood Attack' *exploits* (relationship) a TCP/IP 3-way handshake vulnerability (implied concept) and *threatens* (relationship) the Web server (implied concept).

Cascading Threats. Countermeasures themselves may have threats whereby the entity that protects another, is itself vulnerable. Cyclic dependencies between disparate trees cannot be explicitly modelled using threat tree constraints. From Figure 1(a), installing a firewall ('Firewall-1') with policy configuration 'Syn-Threshold-1A' and 'Syn-Proxy-1' will mitigate or reduce the threat of a 'Syn-Flood Attack' on the Web server. The 'Syn-Proxy-1' countermeasure in effect shifts the threat posed to the the Web server onto the firewall itself and thus the 'Syn-Flood Attack' has now indirectly migrated to 'Firewall-1' giving rise to the threat tree outlined in Figure 1(b). One of the ways that 'Firewall-1' can be protected is for 'Firewall-2' (implicitly defined asset) to filter traffic via its 'Syn-Threshold-2B' and 'Syn-Proxy-2' policy, thereby giving rise to the implicit dependency cycle.

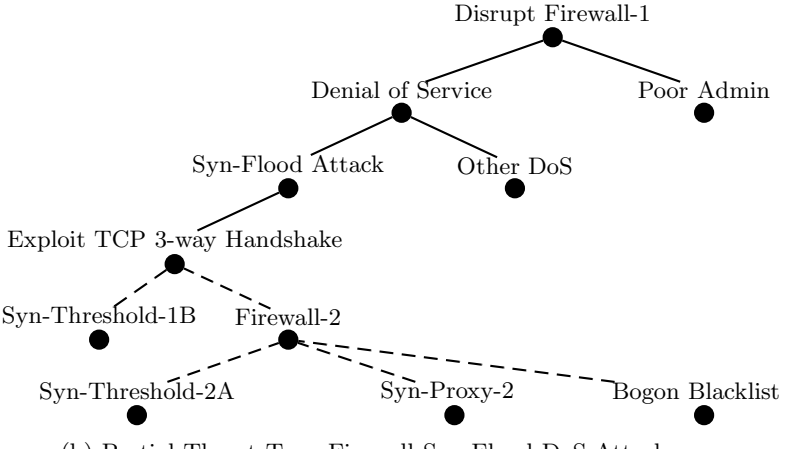
Unclear Threat Hierarchy. Threat trees are typically developed in isolation, that is they focus on a single threat target (decompose 'Disrupt Web Server' threat), and as a consequence it becomes difficult to inter-relate implicit information across multiple threat trees. Both 'Disrupt Web Server' and 'Disrupt Firewall' (Figure 1) form part of a forest of (implicitly related) threat trees. By the definition of a tree, a node should have one parent in the tree and given that the threat tree structure allows for one type of relationship, subsumption, a problem arises. The question is how should the hierarchy be constructed when assembling the overall tree forest? Either a new root node is created, 'Disrupt Servers' where both 'Disrupt Web Server' and 'Disrupt Firewall' are treated as disjoint siblings or the 'Disrupt Firewall' tree becomes a sub-node of the 'Firewall-1' node within the 'Disrupt Web Server' threat tree. The language provided by threat trees is not rich enough to state explicitly that the former sibling approach should be adopted with the inclusion of a dependency relationship that links the two trees together.

Tree Complexity. Modeling complex system threats can be challenging [5] due to tree-explosion and disparate trees (forest). While this may be addressed by software tools, much of the complexity must be managed implicitly and outside of the tree.

Semi-Formal. The semi-formal nature of a threat tree means that, in practice, any reasoning must be done outside of the tree structure [4]. For example, it would be useful to reason whether the concept of 'Denial of Service' has the same meaning in both trees of Figure 1.



(a) Partial Threat Tree: Web Server Syn-Flood DoS Attack



(b) Partial Threat Tree: Firewall Syn-Flood DoS Attack

Fig. 1. Threat Tree Forest

3 Semantic Threat Graphs

A semantic threat graph is an extended threat tree that addresses the issues discussed in Section 2. A semantic threat graph can be defined as a graph that represents the semantics (meaning) of a threat domain. Intuitively, a semantic threat graph makes explicit the information that is typically implicit in a threat tree.

Figure 2 provides a model of the components in a semantic threat graph and Figure 3 depicts an instantiation of this model for the threat tree example in the previous section. Enterprise IT assets are represented as *instances* of the *Asset* concept. An asset may have one or more *hasWeakness*'s (property relationship) that relate to individuals categorised in the *Vulnerability* concept (Figure 2). Instances of the *Vulnerability* concept are exploitable (*exploitedBy*) by a threat or set of threats (*Threat* concept). As a consequence, an asset that has a vulnerability is therefore also *threatenedBy* a corresponding *Threat*. A Countermeasure *mitigates* particular vulnerabilities. Countermeasures are deemed to be kinds-of assets, thus are defined as a *subConceptOf Asset*. Note, the *subConceptOf* has a double arrow head and defines a subsumption relation.

Explicit Concepts & Relationships. Domain experts explicitly specify the concepts (set of instances) of the threat domain and characterise their relationships to other instances and/or concepts. For example, a Network Access Controls expert specifies firewall configurations that adequately mitigates threats identified by a security manager.

Cascading Threats. With the graph-based approach cascading threats are identified explicitly within the threat graph model. Figure 3 demonstrates a threat that cannot be easily represented within a threat tree structure. Asserted relationships (for example *protects*) define that the *web server* is protected by *firewall-1* which, in turn, is protected by *firewall-2*, thus identifying the cascade threat. For simplicity, the scenario shown in Figure 3 models the *3-way handshake* as the

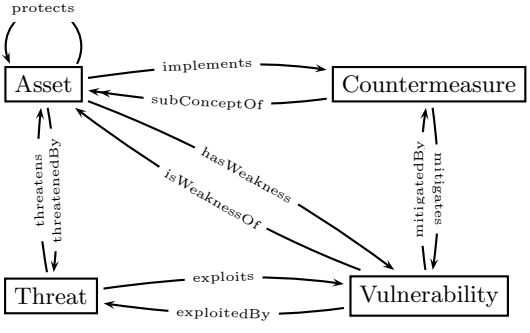


Fig. 2. Abstract Threat Graph Model

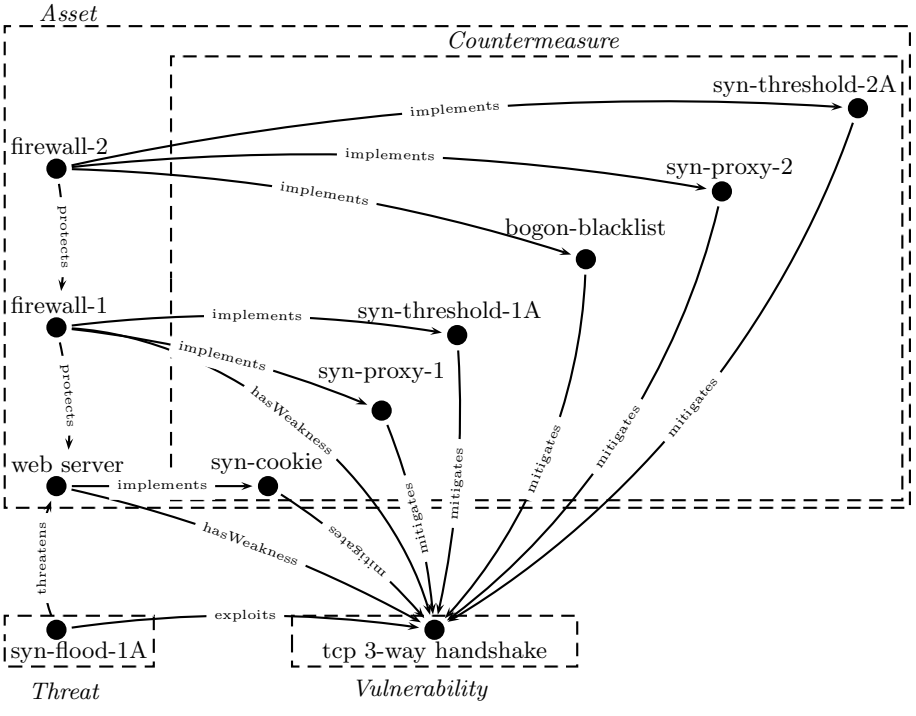


Fig. 3. Threat Graph: Web Server & Firewall Syn-Flood Countermeasure Dependency

same vulnerability for firewalls and systems. As a result of *firewall-1* mitigating the *syn-flood-1A* threat on the *web server* by way of a *syn-proxy-1*, it then adopts that threat while proxying the web servers TCP stream vulnerability.

Taxonomic Graph Hierarchies. Although threat trees provide hierarchies of the *Threat* concept, it lacks the capability to define a hierarchy for the implicit concepts within the tree. The threat graph model presented in Figure 2 can be further refined with sub-concepts that are more refined than their parent concepts. For example, the *Asset* concept can define a sub-concept *Server* to represent the set of servers (instances) an enterprise might have. This concept can in turn be further categorised as *Business Server* (containing Web, Email, Application servers and so forth) and *Protection Server* (for example, Firewalls, IDS's, VPN's, Anti-Spam). The *Threat* concept, as an additional example, can define a number of sub-concepts in accordance with best practice such as the Microsoft STRIDE standard (an acronym) whereby threats are categorised as follows: *S*poofing identity, *T*ampering with data, *R*epudiation, *I*nformation disclosure, *D*enial of service and *E*levation of privilege [16].

Managing Graph Complexity. Like threat trees, semantic threat graphs are prone to excessive graph complexity due to a large number of concept relationships.

The ability to compute a taxonomic hierarchy for a graph can assist navigation of complex graphs. Additionally, semantic reasoners and tools such as *Protégé* [17] help manage, navigate and ensure verifiable consistent graphs.

Formal Semantics. The concepts and relationships in semantic threat graphs are formally specified in terms of an ontology using Description Logic (*DL*) [8]. This is a decidable portion of first-order logic and is well-suited to the representation of and reasoning about domain knowledge.

4 An Ontology for Semantic Threat Graphs

An ontology provides a conceptual model of a domain of interest [7]. It does so by providing a vocabulary describing various aspects of the domain of interest and provides a rich set of constructs to build a more meaningful level of knowledge. Figure 2 depicts an abstract model for semantic threat graphs, whereby *classes* (concepts) represent sets of individuals (instances) and *properties* (roles) represent binary relations applied to individuals. Note that in presenting the model components, for reasons of space, we do not provide complete specifications in particular, definitions do not include disjoint axioms, sub-properties, data type properties or closure axioms.

Asset. Class *Asset* represents any entity of interest within the enterprise that may be the subject of a threat. While assets can include people and physical infrastructure, in this paper we consider computer-system based entities such as Web servers, firewalls, databases, and so forth.

Individuals of class *Asset* may have zero or more vulnerabilities ($\forall$ restriction) along property *hasWeakness*. As a result, those assets may be exposed to various individuals of the *Threat* class. An asset *may* have the capability to implement a countermeasure to protect itself or other assets.

$$Asset \sqsubseteq \forall hasWeakness.Vulnerability \sqcap \forall threatenedBy.Threat \sqcap \forall implements.Countermeasure$$

For example, an instance `webServer` of the *Asset* class that is vulnerable to a syn-flood based Denial of Service (*DoS*) attack can be defined by the following *DL* assertion (A-box), representing a fragment of knowledge in the ontology. Note atomic individuals are written in a `typewriter` font.

$$Asset(\texttt{webServer}) \leftarrow hasWeakeness(\texttt{webServer}, \texttt{tcpHandshake}) \sqcap threatenedBy(\texttt{webServer}, \texttt{synFlood})$$

Class *Asset* can be further sub-classed, to more specific kinds of asset concepts, for example, $ProtectionServer \sqsubseteq Server$, whereby *Server* is a sub-class of *Asset*. The class *ProtectionServer* represents the *NAC* system individuals within the network (Cisco Pix, Linux Netfilter and so forth) and have a role in protecting

(*protects*) internal servers (including themselves). The *ProtectionServer* definition further restricts the *implements* property by requiring a protection server to implement one or more ($\exists_{\geq 1}$) countermeasures.

$$ProtectionServer \sqsubseteq Server \sqcap \exists_{\geq 1} protects.Server \sqcap \exists_{\geq 1} implements.Countermeasure$$

For example, the following fragment of knowledge in the ontology defines that the perimeter firewall (*gatewayNAC*) protects the Web server from a DoS by implementing a syn-threshold filtering countermeasure.

$$ProtectionServer(\texttt{gatewayNAC}) \leftarrow protects(\texttt{gatewayNAC}, \texttt{webServer}) \sqcap implements(\texttt{gatewayNAC}, \texttt{synThres})$$

Threat. A threat is a potential for violation of security [15]. An individual of the *Threat* class is considered to exploit one or more vulnerabilities ($\exists_{\geq 1}$ restriction):

$$Threat \sqsubseteq \exists_{\geq 1} exploits.Vulnerability \sqcap \exists_{\geq 1} threatens.Asset$$

For example, the DoS threat `synFlood` threatens a DMZ `webServer`.

$$Threat(\texttt{synFlood}) \leftarrow exploits(\texttt{synFlood}, \texttt{tcpHandshake}) \sqcap threatens(\texttt{synFlood}, \texttt{webServer})$$

Vulnerability. A vulnerability is a flaw or security weakness in an asset that has the potential to be exploited by a threat.

$$Vulnerability \sqsubseteq \exists_{\geq 1} isExploitedBy.Threat \sqcap \exists_{\geq 1} isWeaknessOf.Asset$$

For example, mis-configured *NAC* configurations have the potential to expose both internal servers and *NAC*'s alike to threats. The following fragment in the ontology states that the `webServer` is susceptible to a `synFlood` attack via the weakness `tcpHandshake`.

$$Vulnerability(\texttt{tcpHandshake}) \leftarrow isExploitedBy(\texttt{tcpHandshake}, \texttt{synFlood}) \sqcap isWeaknessOf(\texttt{tcpHandshake}, \texttt{webServer})$$

Countermeasure. A countermeasure is an action or process that mitigates vulnerabilities and prevents and/or reduces threats.

$$Countermeasure \sqsubseteq Asset \sqcap \exists_{\geq 1} mitigates.Vulnerability$$

Countermeasures can be further sub-classed into specific concepts, if desired. For example, an *AccessCtrlPolicy* is a countermeasure configured as one or more *NACPolicy* rules.

$$AccessCtrlPolicy \sqsubseteq Countermeasure \sqcap \exists_{\geq 1} configuredAs.NACPolicy$$

Countermeasure `synThres` mitigates the vulnerability `tcpHandshake` on the Web server (`webServer`); this countermeasure is configured as a collection of device-specific (Netfilter) *NACPolicy* rules, `synDoSLimit` and `synDoSDrop`.

$$\begin{aligned} AccessCtrlPolicy(\texttt{synThres}) \leftarrow\ & mitigates(\texttt{synThres}, \texttt{tcpHandshake}) \sqcap \\ & configuredAs(\texttt{synThres}, \texttt{synDoSLimit}) \sqcap \\ & configuredAs(\texttt{synThres}, \texttt{synDoSDrop}) \end{aligned}$$

An example of a low-level Netfilter syn-threshold rule-set that: a) limits the number of TCP connections to the web server to 1 per second after 4 connections have been observed and b) offending packets that exceed the limit are dropped, is expressed as follows:

"iptables -A FORWARD -d WebServerIP -p tcp –syn -m limit –limit 1/s –limit-burst 4 -j ACCEPT"
"iptables -A FORWARD -d WebServerIP -p tcp –syn -j DROP"

Based on previous research [9,10], the following are *DL* fragments that are representative of the above Netfilter rules:

$$\begin{aligned} NetfilterRule(\texttt{synDoSLimit}) \leftarrow\ & hasChain(\texttt{synDoSLimit}, \texttt{forward}) \sqcap \\ & hasDstIP(\texttt{synDoSLimit}, \texttt{webServerIP}) \sqcap \\ & hasProtocol(\texttt{synDoSLimit}, \texttt{tcp}) \sqcap \\ & hasTCPFlag(\texttt{synDoSLimit}, \texttt{syn}) \sqcap \\ & hasLimit(\texttt{synDoSLimit}, 1) \sqcap \\ & hasLimitBurst(\texttt{synDoSLimit}, 4) \sqcap \\ & hasTarget(\texttt{synDoSLimit}, \texttt{accept}) \end{aligned}$$

$$\begin{aligned} NetfilterRule(\texttt{synDoSDrop}) \leftarrow\ & hasChain(\texttt{synDoSDrop}, \texttt{forward}) \sqcap \\ & hasDstIP(\texttt{synDoSDrop}, \texttt{webServerIP}) \sqcap \\ & hasProtocol(\texttt{synDoSDrop}, \texttt{tcp}) \sqcap \\ & hasTCPFlag(\texttt{synDoSDrop}, \texttt{syn}) \sqcap \\ & hasTarget(\texttt{synDoSDrop}, \texttt{drop}) \end{aligned}$$

Threshold. This value is used to define the minimum degree of effectiveness of a countermeasure in mitigating the impact of a threat on an asset.

$$Countermeasure \sqsubseteq \exists_{=1} minEffect.Threshold$$

Simlarly, each threat has a threat level that defines the maximum impact on the asset.

$$Threat \sqsubseteq \exists_{=1} maxImpact.Threshold$$

For example, *Threshold* can be defined as enumerated class:

$$Threshold \sqsubseteq \{\texttt{high,medium,low,nil}\}$$

and a fragment in the ontology is

$$\begin{aligned} Threat(\texttt{synFlood}) \leftarrow\ & exploits(\texttt{synFlood}, \texttt{tcpHandshake}) \sqcap \\ & threatens(\texttt{synFlood}, \texttt{webServer}) \sqcap \\ & maxImpact(\texttt{synFlood}, \texttt{high}) \end{aligned}$$

The threshold level is used to characterize the extent to which a countermeasure mitigates a threat: an asset is considered secure if the effectiveness (threshold) of the countermeasures are greater than the impact (threshold) of the related threats. For example, if a `synCookie` was considered to have `medium` effectiveness at mitigating a `synFlood` then the asset remains under threat, albeit less threat than having no countermeasure. While CVSS [18], CVE [19], DREAD [20] and OSVDB [21] for example may provide suitable threshold metrics, the elicitation of threshold weightings is not the focus of this paper.

5 Case Study

5.1 Network System Configuration

A simplified 3-tier e-commerce *NAC* architecture is illustrated in Figure 4. The network at tier-1, also known as a Demilitarized Zone (*DMZ*), hosts the Web server that is accessible from the Internet. The gateway *NAC*, a firewall, implements a configuration that permits inbound packets from the Internet to the Web server on ports HTTP and HTTPS by way of an *AccessCtrlPolicy* $\texttt{cntr}_{web}$ countermeasure and drops all other irrelevant packets ($\texttt{cntr}_{denyOtherPkt}$). The following is a fragment of knowledge that defines the gateway firewall:

$$\begin{aligned} ProtectionServer(\texttt{gatewayNAC}) \leftarrow\ & protects(\texttt{gatewayNAC}, \texttt{webServer}) \sqcap \\ & implements(\texttt{gatewayNAC}, \texttt{cntr}_{web}) \sqcap \\ & implements(\texttt{gatewayNAC}, \texttt{cntr}_{denyOtherPkt}) \end{aligned}$$

Within tier-2, the application server communicates with the Web server over an SSL tunnel. The application firewall, (`appNAC`), implements countermeasure $\texttt{cntr}_{sslTunnelApp}$ to permit the correct SSL access.

$$\begin{aligned} ProtectionServer(\texttt{appNAC}) \leftarrow\ & protects(\texttt{appNAC}, \texttt{appServ}) \sqcap \\ & implements(\texttt{appNAC}, \texttt{cntr}_{sslTunnelApp}) \sqcap \\ & implements(\texttt{appNAC}, \texttt{cntr}_{denyOtherPkt}) \end{aligned}$$

Hosted in tier-3 is the database server. The backend data *NAC* is configured to permit SSH traffic, ($\texttt{cntr}_{sshTunnelDB}$), from the application server destined for the database server only.

$$\begin{aligned} ProtectionServer(\texttt{dataNAC}) \leftarrow\ & protects(\texttt{dataNAC}, \texttt{dbServer}) \sqcap \\ & implements(\texttt{dataNAC}, \texttt{cntr}_{sshTunnelDB}) \sqcap \\ & implements(\texttt{dataNAC}, \texttt{cntr}_{denyOtherPkt}) \end{aligned}$$

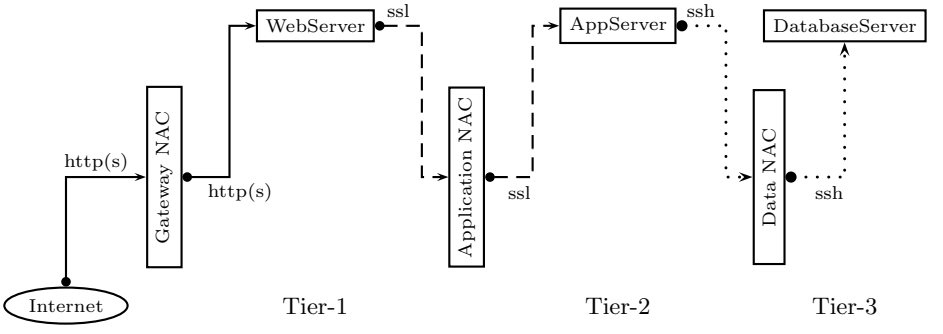


Fig. 4. Abstract 3-Tier Enterprise E-Commerce *NAC* Architecture

5.2 Threshold Satisfiability Analysis

The ideal *NAC* configuration is one that permits only valid traffic, and, no more and no less. An example of threats to the database, asserted within the ontology, is identified in Table 1. The focus here is on the threat of permitting a set of clients (IP Addresses) direct or indirect access to the database. For example, whether the database be accessible directly by the Web server IP address, in which case this threat impact is considered `high` as identified by $\texttt{threat}_{web}$. Note that $\texttt{threat}_{tier1}$ subsumes $\texttt{threat}_{web}$ since the Web server's IP address is contained in the network IP range of tier-1.

Table 1. Example Threats of Unintended IP Address Access to the Database

Asset	Threat	Unintended IP Access	$\text{Threat}_{maxImpact}$
dbServer	threat_{web}	webServIP	high
dbServer	threat_{tier1}	tier1Subnet	high
dbServer	threat_{tier2}	tier2Subnet	medium
dbServer	threat_{app}	appServerIP	nil

The following *DL* fragment represents the multiple *NAC* systems protecting the database server as part of a defense-in-depth strategy:

$$\begin{aligned} \mathit{DatabaseServer}(\texttt{dbServer}) \leftarrow\ & \mathit{isProtectedBy}(\texttt{dbServer}, \texttt{gatewayNAC}) \\ & \mathit{isProtectedBy}(\texttt{dbServer}, \texttt{appNAC}) \sqcap \\ & \mathit{isProtectedBy}(\texttt{dbServer}, \texttt{dataNAC}) \end{aligned}$$

A sample collection of *NAC* policy configurations (defense-in-depth) currently protecting the database is provided in Table 2. We consider a number of inadequate countermeasures that expose the database to unintended access. The table outlines the name of the *NAC* system, its implemented countermeasure

Table 2. Example NAC Policy Countermeasures and their associated Effectiveness

NAC	NAC_{cntr}	$cntr_{minEffect}$	NAC_{rule}	Src IP	Dst IP	DstPort
appNAC	$cntr_{webTrafficGen}$	low	$nacr_{app1}$	webServIP	tier2Subnt	any
dataNAC	$cntr_{sshTunnelDB}$	high	$nacr_{bac1}$	appServIP	dbServIP	22
dataNAC	$cntr_{sshTunnelDBGen}$	medium	$nacr_{bac2}$	tier2Subnet	dbServIP	22
dataNAC	$cntr_{sshTier1Gen}$	low	$nacr_{bac3}$	tier1Subnet	dbServIP	22

(NAC_{cntr}), the mitigation effectiveness ($cntr_{minEffect}$) and the corresponding low-level NAC_{rule} components (such as source IP address).

The `appNAC` firewall implements a generic access countermeasure that enables much more than the intended SSL access to the application server. In effect, $\texttt{cntr}_{webTrafficGen}$ permits the Web server access to all systems and services in tier-2 and is rated as having a `low` effectiveness in mitigating unnecessary database access. A compromised Web server can now be used as a launch pad for attacks on systems in tier-2, inclusive of the `dataNAC` Ethernet interface within that network tier. A more restrictive countermeasure ($\texttt{cntr}_{sslTunnelApp}$) is required.

The correctly implemented countermeasure ($\texttt{cntr}_{sshTunnelDB}$) within the `dataNAC` policy configuration helps in mitigating the misconfiguration of firewalls upstream. Thus defence-in-depth is borne out as a result. However, in practice, managing *NAC* policy configurations is complex [9] and there may be a requirement for multiple application servers within tier-2 to communicate with the database server. Rather than defining specific countermeasures for each application server, an administrator may, in the knowledge of being protected by a firewall upstream, implement a generic countermeasure such as $\texttt{cntr}_{sshTunnelDBGen}$, that provides blanket SSH access to the database server from all systems in tier-2. Other non-bastion hardened servers (such as intranet LDAP server) in tier-2 could then be used as launch pad when attacking the database server in tier-3, for example a SSH brute force attack. On individual basis both $\texttt{cntr}_{webTrafficGen}$ and $\texttt{cntr}_{sshTunnelDBGen}$ may appear as minor oversights in their respective tiers, however it is their conjunction that provides indirect and unacceptable access from the Web server to the database.

The `dataNAC` firewall inadequately mitigates the threat $\texttt{threat}_{tier1}$ (outlined in Table 1), as countermeasure $\texttt{cntr}_{sshTier1Gen}$ directly permits SSH access from all tier-1 systems to the database. Perhaps remote database administration is a requirement through an SSH proxy server in the DMZ. Unintended access from the tier-1 through the `dataNAC` may be considered to have less of a threat impact if one can guarantee that firewalls upstream apply more restrictive port-forwarding controls. However, and in keeping with the defense-in-depth approach, countermeasure $\texttt{cntr}_{sshTier1Gen}$ should be refined to have a more restrictive access policy.

Analysis of security policy configuration is performed using the Semantic Web Rule Language *SWRL* [22]. *SWRL* complements *DL* providing the ability to infer additional information from *DL* constrained ontologies. *SWRL* rules are

Table 3. Inadequate Countermeasure Analysis Report

Threat	$\text{Threat}_{maxImpact}$	NAC	$\text{NonCompliantNAC}_{cntr}$	$\text{cntr}_{minEffect}$	NAC_{rule}
threat_{web}	high	appNAC	$\text{cntr}_{webTrafficGen}$	low	nacr_{app1}
threat_{tier1}	high	dataNAC	$\text{cntr}_{sshTier1Gen}$	low	nacr_{bac3}

Horn-clause like rules written in terms of *DL* concepts, properties and individuals. A *SWRL* rule is composed of an antecedent (body) part and a consequent (head) part, both of which consist of positive conjunctions of atoms.

The following is an excerpt of a *SWRL* query ($sqwrl : select$) that analyses the overall *NAC* configuration for breaches in threshold satisfiability due to inadequate countermeasures. It reports if countermeasures ($?c$) implemented by a *NAC* ($?nac$) to protect vulnerable assets ($?a$) are not effective ($swrlb : lessThan$ comparison) in mitigating the threat impacts ($maxImpact$).

$$threatenedBy(?a, ?t) \wedge hasWeakness(?a, ?v) \wedge exploits(?t, ?v) \wedge mitigates(?c, ?v) \wedge$$
$$isProtectedBy(?a, ?nac) \wedge implements(?nac, ?c) \wedge maxImpact(?t, ?imp) \wedge$$
$$minEffect(?c, ?eff) \wedge \ldots \wedge swrlb : lessThan(?eff_{value}, ?imp_{value})$$
$$\rightarrow sqwrl : select(?t, ?imp, ?nac, ?c, ?eff, ?nr)$$

Table 3 depicts the results of this query on the knowledge-base regarding inadequate threat mitigation of unintended client access.

5.3 Configuration Recommendation Synthesis

Synthesis provides configuration recommendations from a catalogue of best-practice countermeasures. For example, it is considered best practice that *NAC*'s implement anti-spoofing bogon rules as described by [11,12,13] to protect its internal servers and end-user work stations. As a consequence, a *NAC* policy should prohibit incoming packets claiming to originate from the internal network. Similarly both [6,23] are examples of best practice with regard to mitigating DoS attacks. Countermeasures `synDoSLimit` and `synDoSDrop` introduced in Section 4 are examples of DoS catalogue countermeasures.

The following *SWRL* rule excerpt, states if an asset is threatened by a spoofing attack (`spoof` individual) as a result of a particular weakness in the TCP/IP stack, `ipHeaderForgery` (individual) then that asset's protecting *NAC* should implement a set of best practice catalogue countermeasures that adequately ($swrlb : greaterThanOrEqual$) reduce the threat.

$$threatenedBy(?a, \texttt{spoof}) \wedge hasWeakness(?a, \texttt{ipHeaderForgery}) \wedge$$
$$exploits(\texttt{spoof}, \texttt{ipHeaderForgery}) \wedge mitigates(?c, \texttt{ipHeaderForgery}) \wedge$$
$$isProtectedBy(?a, ?nac) \wedge maxImpact(\texttt{spoof}, ?imp) \wedge minEffect(?c, ?eff) \wedge$$
$$\ldots \wedge swrlb : greaterThanOrEqual(?eff_{value}, ?imp_{value}) \rightarrow implements(?nac, ?c)$$

6 Tool Support

The *semantic threat graph* (ontology) and case study described in this paper where implemented in OWL-DL, a language subset of OWL which is a W3C standard that includes *DL* reasoning semantics [24]. *Protégé* is a plug-and-play knowledge acquisition framework that provides a graphical ontology editor [17]. Protégé interfaces with a *DL* based reasoner called *Pellet* providing model classification and consistency [25]. In conjunction to *DL* reasoning support, the *SWRL* Protégé plug-in (SWRLTab), allows for the creation of horn-like logic rules that interfaces with an expert system called *Jess* [26,22]. In practice, a domain expert using such tools can avoid or at least limit having to become an expert in *DL* and/or *OWL* notation, as these semantic editors and underlying reasoning tools hide much of the underlying complexity.

7 Related Research

A number of existing research approaches extend the threat tree model in particular [4,27,5,28]. Additional boolean node operators: *NAND, XOR and NOR*, and the incorporation of *defense nodes* as countermeasures is described by [28]. Edge et al [5] define a Protection Tree and Bistarelli et al [4] define a Defense Tree as countermeasure-centric extensions to the threat tree approach. The research carried out by [27] describes an *Enhanced Attack Tree (EAT)* that supports temporal dependencies and sequential threat events that must occur for an attack to be successful.

While these approaches are aimed at resolving particular inadequacies within the vanilla threat tree model, they still operate at rather high-levels of abstraction and are limited with regard to viable threat-only-vectors that contain implicit information such as assets, vulnerabilities and so forth. Our approach differs by extending the threat tree model to include semantic knowledge about fine-grained security configuration and, how it relates to assets, threats, vulnerabilities and countermeasures. Thus, the graph based approach makes explicit the information that is typically implicit in a threat tree. The semantic threat graph model is implemented in a knowledge representation system (Protégé) that provides a language for defining an ontology and an ability to conduct inferences accross the graph.

While the semantic threat graph model proposed in this paper is applicable to any threat and countermeasure domain, the case study focused on NAC countermeasures, building on existing ontologies [9,10]. An ontology for Netfilter firewall is described in [10] and is used to (binary) test the consistency of firewall rules with respect to a Semantic Web application policy. [9] considers the inter-operation of multiple Netfilter firewall and TCPWrapper proxies with respect to business and network service requirements.

8 Conclusion

This paper outlined a threat management approach using an ontology to construct, reason about and manage security policy configurations in the context of semantic threat graphs. Threat tree models used to represent threats at a high-level of abstraction, their singular threat vector focus and their practical suitably in a localised context (individual trees) do not explicitly capture all the entities involved in the threat management process. The *STG* extends the threat tree model to include semantic knowledge about low-level security configurations.

The model was used to build a knowledge-base of best-practice countermeasures (e.g. PCI-DSS & NIST) and system security policies against known threats. The ontology was populated with around one hundred threat, vulnerability and countermeasure combinations providing a relatively small catalogue for testing purposes. A case study on *NAC* demonstrated how security configurations can be analysed using knowledge about the countermeasures effectiveness in mitigating threats and how automated security mechanism configuration recommendations can be made based on catalogues of best-practice countermeasures.

Future research shall investigate how larger catalogues might be pre-populated from existing vulnerability databases such as OSVDB using knowledge engineering techniques such as case-based reasoning. An investigation of scalability regarding our approach will also need to be conducted. Emphasis on dynamic elicitation of threshold weightings also needs to be considered.

Acknowledgments. This research has been supported by Science Foundation Ireland grant 08/SRC/11403.

References

1. Mauw, S., Oostdijk, M.: Foundations of Attack Trees. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer, Heidelberg (2006)
2. Stamatelatos, M., Vesely, W., Dugan, J., Fragola, J., Minarick, J., Railsback, J.: Fault Tree Handbook with Aerospace Applications. NASA Office of Safety and Mission Assurance NASA Headquarters, Washington, DC 20546, Version 1.1 (August 2002)
3. Schneier, B.: Secrets and Lies Digital Security in Networked World. Wiley Publishing, Chichester (2004)
4. Bistarelli, S., Fioravanti, F., Peretti, P.: Defense trees for economic evaluation of security investments. In: 1st International Conference on Availability, Reliability and Security (ARES), Vienna (April 2006)
5. Edge, K., Raines, R., Grimaila, M., Baldwin, R., Bennington, R., Reuter, C.: The Use of Attack and Protection Trees to Analyze Security for an Online Banking System. In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS 2007) (2007)
6. Eddy, W.: RFC 4987: TCP SYN Flooding Attacks and Common Mitigations (August 2007), `http://ietf.org`
7. Taniar, D., Rahayu, J.W.: Web Semantics Ontology. Idea Publishing (2006)

8. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, Cambridge (2003)
9. Fitzgerald, W.M., Foley, S.N., Foghlú, M.O.: Network Access Control Interoperation using Semantic Web Techniques. In: 6th International Workshop on Security In Information Systems (WOSIS), Barcelona, Spain (June 2008)
10. Foley, S.N., Fitzgerald, W.M.: Semantic Web and Firewall Alignment. In: First International Workshop on Secure Semantic Web (SSW 2008), Cancun, Mexico. IEEE CS Press, Los Alamitos (2008)
11. IANA: RFC 3330: Special-Use IPv4 Addresses (September 2002), `http://ietf.org`
12. Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., Lear, E.: RFC1918: Address Allocation for Private Internets (Febuary 1996), `http://ietf.org`
13. Wack, J., Cutler, K., Pole, J.: Guidelines on Firewalls and Firewall Policy: Recommendations of the National Institute of Standards and Technology. NIST-800-41 (2002)
14. Tracy, M., Jansen, W., Scarfone, K., Winograd, T.: Guidelines on Securing Public Web Servers: Recommendations of the National Institute of Standards and Technology. NIST Special Publication 800-44, Version 2 (September 2007)
15. Shirey, R.: RFC 2828: Internet Security Glossary (May 2000), `http://ietf.org`
16. Hernan, S., Lambert, S., Ostwald, T., Shostack, A.: Uncover Security Design Flaws Using The STRIDE Approach (2009), `http://microsoft.com/`
17. Gennari, J., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubezy, M., Eriksson, H., Noy, N.F., Tu., S.W.: The Evolution of Protege: An Environment for Knowledge-Based Systems Development. Journal of Human-Computer Studies 58(1) (2003)
18. FIRST: Common Vulnerability Scoring System (2009), `http://first.org/cvss/`
19. International, C.: Common Vulnerabilities and Exposures (2009 `http://cve.mitre.org/`
20. Meier, J., Mackma, A., Dunner, M., Vasireddy, S., Escamilla, R., Murukan, A.: Improving Web Application Security: Threats and Countermeasures. Microsoft Press (2003)
21. OSVDB: Open Source Vulnerability Database (2009), `http://osvdb.org/`
22. O'Connor, M.J., Knublauch, H., Tu, S.W., Grossof, B., Dean, M., Grosso, W.E., Musen, M.A.: Supporting Rule System Interoperability on the Semantic Web with SWRL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 974–986. Springer, Heidelberg (2005)
23. Ferguson, P.: RFC 2827: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing (May 2000), `http://ietf.org`
24. Smith, M.K., Welty, C., McGuinness, D.L.: OWL Web Ontology Language Guide. W3C Recommendation, Technical Report (2004)
25. Parsia, B., Sirin, E.: Pellet: An OWL DL Reasoner. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298. Springer, Heidelberg (2004)
26. Friedman-Hil, E.J.: Jess the Rule Engine for the Java Platform. Version 7.0p1 (2006)
27. Camtepe, S.A., ulent Yener, B.: Modeling and Detection of Complex Attacks. In: 3rd International Conference on Security and Privacy in Communications Networks, Secure Comm, Nice, France (September 2007)
28. Opel, A.: Design and Implementation of a Support Tool for Attack Trees. Internship Thesis, Otto-von-Guericke University Magdeburg (March 2005)

Reaction Policy Model Based on Dynamic Organizations and Threat Context

Fabien Autrel, Nora Cuppens-Boulahia, and Frédéric Cuppens

Telecom-Bretagne, 35576 Cesson Sévigné (France)

Abstract. The tasks a system administrator must fulfill become more and more complex as information systems increase in complexity and connectivity. More specifically, the problem of the expression and update of security requirements is central. Formal models designed to express security policies have proved to be necessary since they provide non ambiguous semantics to analyze them. However, such models as RBAC or OrBAC are not used to express reaction requirements which specify the reaction policy to enforce when intrusions are detected. We present in this article an extension of the OrBAC model by defining dynamic organizations and threat contexts to enable the expression and enforcement of reaction requirements.

1 Introduction

Information systems are becoming more and more complex and system administrators have to face several problems. In this context, specifying a security policy becomes a very tedious and error prone task. Using a formal approach brings several benefits: the policy expression is non-ambiguous and tools can be used to analyze a security policy[CCBG07] and deploy it[PCBC+07].

A security policy may express very different requirements and can contain different types of policies: authentication, access control, flow control and usage control requirements. However, the reaction policy, which expresses the security requirements related to the detection of attacks, is generally not considered as part of the security policy. This reaction policy expresses which reaction requirements (RR) should be enforced when the security policy is violated.

Specifying and deploying RR is actually not an easy task. [DTCCB07] is a preliminary work in this direction but we shall explain why this approach is not fully satisfactory. In this paper, we shall define a model based on the concept of dynamic organization created to manage intrusions which significantly enhances the approach suggested in [DTCCB07].

In this paper we suggest to express the RR using the OrBAC model. Section 2 presents the problems related to the expression of the RR. Section 3 reviews some related work. Section 4 outlines the OrBAC model and section 5 explains how we use it to model the RR. Section 6 presents three examples of RR related to an elementary attack and two multi-step intrusions. Section 7 presents the implementation in the MotOrBAC 2[1] support tool. We conclude in section 8.

[1] http://motorbac.sourceforge.net

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 49–64, 2009.

2 Problematic

In this paper, the task of expressing the RR takes place into a supervision loop (figure 1) where the processes of intrusion detection and policy specification/enforcement interact with each other. In this approach we consider that the reaction module implements a policy-based reaction process, i.e, upon the detection of an attack, the reaction process consists in enforcing new security rules.

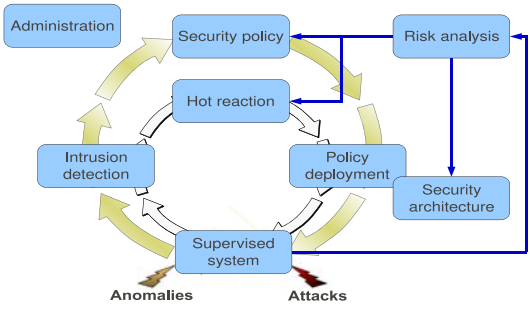


Fig. 1. Supervision loop

The RR expressed in a reaction policy may include prohibitions, obligations and permissions. Those requirements can be expressed as a set of reaction rules which are specified for each attack that the intrusion detection systems may detect. For example the informal specification of the RR for a brute force attack executed against a UNIX account of a SSH server could be the following requirements:

1. Block the attacker
2. Prevent the attacker from getting a user access to the target computer
3. Suspend the victim account
4. Warn the user who owns the attacked account and tell him he must change his/her password

We shall show that some requirements can be formalized as obligations (requirements 1, 3, 4) and prohibitions (requirement 2). Additionally, those requirements should be only enforced in the context of a brute force attack. Note that the fourth requirement can be divided into two requirements: the user must be informed that his account has been compromised and once he has been warned, he must change his password. In the following section, we show that the policy model which should be used to express the RR must satisfy several properties.

Formalizing the reaction policy. Let us consider a set of Attacks A. Writing the RR for an attack A_i in A consists in specifying a set of security rules R_i which are activated when the attack A_i is detected.

The notion of role introduced in the RBAC model is mandatory to specify RR since we cannot know in advance the ip addresses and/or identity of the attackers and victims. However, RBAC is not sufficiently expressive to express RR for several reasons:

- The Attacker and Victims RBAC roles names must be unique for each A_i. Actually if two rules $r \in R_1$ and $r' \in R_2$ specified for attacks A_1 and A_2 use the same role *attacker*, a problem appears when the set of rule R_1 must be activated because attack A_1 has been detected. Indeed if the attacker of A_1 instance is assigned to role *attacker*, r will be activated and r' too despite the fact that no occurrence of A_2 has been detected.
- Since the original RBAC model specifies that user-role assignment is handled statically [FSG+01], the rules R_i for each attack A_i cannot be dynamically activated. Even if considering a mechanism which dynamically assigns alert subjects to roles when attacks are detected, the model lacks expressiveness to associate each R_i to a given A_i.
- The concept of session in the RBAC model could be used to assign subjects to roles when an attack is detected, but only one subject can be assigned to a session. This raises a problem for distributed attacks involving several attackers and victims.
- The dynamic nature of a reaction policy cannot be modeled with RBAC. For example the set of rules R_i associated with attack A_i should be activated when the attack is detected, but at least a subset of those rules should be deactivated after the end of the attack. This could possibly be modeled through the use of several roles, but this adds even more artificial roles.

An extended RBAC model including contexts [MF03] can express the link between an alert classification and a rule through the use of security context if this context is activated when a given intrusion is detected. However, as said above, it is impossible to associate activated rules with the corresponding occurrence of an attack. Moreover the meaning of the context in this extension is that of a condition evaluated when an access is made to the system.

We propose to use the concept of dynamic organization to address the problem of multiple occurrences of an attack. Intuitively, a new dynamic organization is created to manage a given intrusion and different subjects will be assigned locally to roles (like *victim*, *attacker*) within this dynamic organization. Once the intrusion is processed, this dynamic organization is deactivated.

3 Related Work

A majority of research works has focused on the detection of intrusions but few works exist in the field of response to intrusions. Incidentally, very few articles deals with the expression of RR and its enforcement. Some taxonomies of intrusion responses have been defined ([Fis96, CP00, SBW07]). Fisch's taxonomies [Fis96] are system-oriented, they distinguish intrusion response systems by degree of automation and activity of triggered response. Carver and Posh's [CP00]

taxonomy deals with response, but from the attack side. Their taxonomy includes 6 dimensions and does not classifies responses. In [Bra98], Brackney says that detection is the first step leading to a response process and explains that taking action after detection is not a trivial task. Actually reacting quickly and efficiently is difficult considering the time needed to analyze an intrusion, hence an automated and autonomous response system is desirable.

In [CCBB+08, DTCCB07], the OrBAC model is used to express the RR. A new type of OrBAC context is introduced to manage intrusion detection alerts expressed in IDMEF (Intrusion Detection Message Exchange Format)[HCF07]. Such a context specifies the alert classification, an identifier that defines which attack is detected, that triggers its activation and the mapping between alert attributes and concrete entities (subjects, actions and objects). However, the approach lacks a mechanism to specify the mapping between concrete entities mapped with the alert and abstract entities (roles, activities and views) specified in the OrBAC security rules. Moreover the mapping between alert attributes and concrete entities is specified for each context although a more generic mapping that could apply to every threat context would be more convenient and scalable.

In this paper, we propose to use the concept of threat contexts, but the definition we give is different from the one proposed in the aforementioned articles. We separately specify the alert classification which triggers the context activation and the mapping between alert attributes and concrete entities. Abstract entity definition are used to specify this mapping and assign concrete entities to abstract entities. Moreover, we introduce the concept of threat organisations. Hence, we can define intrusion-dependent roles like *attacker* and *victim* and consider that subjects are assigned to these roles, but locally to this threat organization. Thus, it is possible to consider that two different subjects are both assigned to the role *victim* but in two different threat organizations. This will be typically the case if these two subjects are victims of two different intrusions.

4 Introduction to OrBAC

OrBAC [KBB+03] aims at modelling a security policy centered on the concept of organization. An organization defines and manages a security policy. An OrBAC policy specification is done at the organizational level, also called the abstract level, and is implementation-independent. The enforced policy, called the concrete policy, is inferred from the abstract policy.

This approach makes all the policies expressed in the OrBAC model reproducible and scalable. Actually once the concrete policy is inferred, no modification or tuning has to be done on the inferred policy since it would possibly introduce inconsistencies. The inferred concrete policy expresses security rules using subjects, actions and objects. The abstract policy, specified at the organizational level, is specified using *roles*, *activities* and *views* which abstract the traditional *subjects*, *actions* and *objects*.

The OrBAC model uses a first order logic formalism with negation. However, since first order logic is generally undecidable, we have restricted our

model in order to be compatible with a stratified Datalog program [Ull89]. A stratified Datalog program can be evaluated in polynomial time. In the rest of this article the security rules must correspond to a stratified Datalog program. We use a Prolog-like notation where terms beginning with an upper case are variables and terms beginning with a lower case are constants. The fact $parent(john, jessica)$. says that *john* is a parent of *jessica*. A rule such as $grandparent(X, Z) : -parent(X, Y), parent(Y, Z)$. means that X is a grandparent of Z if Y exists such that X is a parent of Y and Y is a parent of Z.

Using this formalism, each organization specifies its own security rules. Some *role* may have the permission, prohibition or obligation to do some *activity* on some *view* given an associated *context* is true. The *context* concept [CCB08] has been introduced in OrBAC in order to express dynamic rules. Those security rules are represented using 5-ary predicates:

- $permission(org, role, activity, view, context)$ means that in organization org, role *role* is authorized to perform activity *activity* on view *view* if context *context* is true.
- the *prohibition* and *obligation* predicates are similarly defined but express different security requirements. The *prohibition* predicate states that a role is not authorized to perform some activity on some view when a given context is true. The *obligation* predicate means that some role *must* do some activity on some view when the associated context is true.

For example, the expression:

permission(hospital, nurse, consult, medical_record, emergency)

means that nurses can access the patients medical records in the context of an emergency. Security rules can be hierarchically structured so that they are inherited in the organization, role, activity and view hierarchies (see [CCBM04]). Since a security policy can be inconsistent because of conflicting security rules (for example a permission can be in conflict with a prohibition), it is necessary to define strategies to solve those conflicts [CCBG07].

Once the security policy has been specified at the organizational level, it is possible to instantiate it by assigning concrete entities to abstract entities. To do so, three ternary predicates have been defined to assign a subject to a role, an action to an activity and an object to a view:

- $empower(org, subject, role)$: specifies that in organization org, subject *subject* is empowered in role *role*.
- $consider(org, action, activity)$: specifies that in organization org, action *action* implements activity *activity*.
- $use(org, object, view)$: specifies that in organization org, object *object* is used in view *view*.

For example, the fact $empower(hospital, john, surgeon)$ states that *john* is empowered in the role *surgeon* in the *hospital* organization.

Contexts are defined through logical rules which express the condition that must be true in order for the context to be active. In the OrBAC model such rules have the predicate *hold* in their conclusion:

– $hold(org, subject, action, object, context)$: specifies that in organization org, subject $subject$ does action $action$ on object $object$ in context $context$.

As suggested in [CCB08], contexts can be combined in order to express conjunctive contexts (denoted &), disjunctive contexts (denoted $\oplus$) and context negation (denoted $\overline{ctx}$, ctx being a context name).

Using this model, concrete security rules applying to subject, actions and objects can be inferred as specified by the following derivation rule:

$$is\_permitted(Subject, Action, Object) :- \\ permission(Org, Role, Activity, View, Context), \\ empower(Org, Subject, Role), \\ consider(Org, Action, Activity), \\ use(Org, Object, View), hold(Org, Subject, Action, Object, Context).$$

Similar rules are defined to infer the *is_prohibited* and *is_obliged* predicates which represent concrete prohibitions and concrete obligations.

5 Reaction Policy Specification

As said in section 2, when an intrusion A_i is detected, a set of security rules R_i should be activated. We showed that modeling a reaction policy in RBAC requires to introduce at least as many roles as attacks and that the link between roles and intrusion detection alerts cannot be established. The concept of organization in the OrBAC model is well-suited to address those problems.

Actually in OrBAC a subject is empowered into a role within an organization. It is possible to define generic intrusion related roles such as *attacker* and *victim* and activate a subset of rules R_i into an organization O_i thanks to the contexts associated with the rules. Those contexts are activated upon the detection of an occurrence of A_i. This way the set of concrete rules inferred from R_i are linked with the occurrence of A_i which activates the contexts used in R_i through the organization O_i. In this approach the dynamic organizations associated with the detection of attack occurrences are dynamically and automatically created instead of being handled manually by an administrator. In the context of intrusion detection, we call *threat organizations* these dynamic organizations. In some sense, there is some analogy with the activation of a session in the RBAC model. However, there are two main differences with a session: an organization threat is automatically created when an alert is received to process the associated attack and there are several subjects involved in an organization threat, typically one or several attackers and one or several victims.

In the remaining of this article we consider that the alerts are expressed in the IDMEF format. The IDMEF format is an object-oriented representation of the data generated by Intrusion Detection Systems (IDSs). It defines several classes such as *Source* and *Target* to represent respectively the attacker and the victim of an event detected by an IDS and the *Classification* class which uniquely identifies the event.

5.1 Threat Organization and Threat Context

We assume that the rules in R making up the RR are defined in an organization called *supervision*. The threat organizations are created as sub-organizations of *supervision* so that the reaction policy is inherited. When a new IDMEF alert is generated, a new threat organization is created to manage it. Let us consider an alert $alert_{ij}$ which is the alert generated upon the detection of the j^{th} occurrence of attack A_i. The following predicate becomes true and binds the alert and the new organization $threat\_org_{ij}$:

$threat\_context\_management(alert_{ij}, threat\_org_{ij})$

We defined in [CBCdV+08], a complete ontological mapping from IDMEF schema onto abstract OrBAC roles, activities and views managed by the supervision organization. Due to space limitation, we do not present this mapping here but simply introduce those abstract entities we use in the remainder of this paper:

- roles *attacker* and *victim*
- activity *attack*
- views *to_victim* and *to_attacker* to additionally describe various information related to the victims and attackers of an intrusion.

We then introduce a new context type (see [CCB08] for other types of contexts) called threat context. A threat context is activated in *threat_org* to manage a given alert (modelled using the predicate *threat_context_management(Alert, Theat_org)*) if its definition matches the classification of the alert (for this purpose we use the predicate $mapping\_classification(Alert, Threat\_context)$). This context is active for every triple $\{subject, action, object\}$ and defined as follows:

$hold(Threat\_org, \_, \_, \_, Threat\_context) :-$
$threat\_context\_management(Alert, Threat\_org),$
$mapping\_classification(Alert, Threat\_context).$

This rule says that, if a given threat organization *threat_org* is associated with the management of an IDMEF alert *alert* and if this alert classification maps onto a threat context *threat_context*, then this threat context is activated in the threat organization *threat_org* for every subject, action and object (denoted by the do not care symbol "_").

Using this context in the specification of rules R_i, or a composition of context including it, allows the activation of the rules in R_i or a subset of R_i in *threat_org*. The activated rules for alert *Alert* are deleted when the threat is no more active by deleting organization *threat_org*.

Notice that in case of conflicts, we consider that security rules that depend on threat contexts have higher priority than other security rules (see [DTCCB07] for more explanation).

5.2 Mapping Alert to Abstract Entities

By contrast to [DTCCB07], The mapping from an alert to the threat abstract entities in *threat_org* is done using role, activity and view definitions. Here is an example of generic mapping between the source of an alert and the *attacker* role:

$$empower(Threat\_org, Subject, victim) :-$$
$$threat\_context\_management(Alert, Threat\_org),$$
$$mapping\_target(Alert, Subject).$$

The mapping from the alert to the abstract entities is independent from the specification of the threat context. This way a reaction policy can be updated by adding new rules corresponding to new threats generally without having to specify the mapping for each new threat.

5.3 Expression of Security Requirements

The reaction policy may express various security requirements as introduced in the example of section 2. Those requirements may include permissions activated in the context of an attack $attack_1$, for example the communication between two servers located in two different networks, which are normally independent, to backup critical data:

$$permission(supervision, data\_server, open\_ssh\_connection, to\_backup\_server, attack_1\_ctx)$$

Here hosts assigned to the *data_server* role are authorized to backup their critical data on hosts assigned to the *to_backup_server* view. This may be implemented by adding rules in some firewalls and routers.

Prohibitions can also be part of the RR. Consider the case of an attacker trying to connect to a telnet server running on a router from outside a company's network. A possible reaction might be to block the traffic coming from the attacker's machine and going to the victim's machine:

$$prohibition(supervision, attacker, all\_traffic, victim, telnet\_attack\_ctx)$$

In this abstract rule, the $all\_traffic$ activity abstracts the network protocols so that there is no need to express the prohibition for each network protocol.

A reaction requirement may be expressed by means of obligations. For instance a web server may be vulnerable to a newly discovered vulnerability and it should be stopped when an attacker tries to exploit this new vulnerability:

$$obligation(supervision, web\_server\_daemon, stop, web\_server, new\_threat)$$

Note that we consider that this obligation is an immediate obligation, i.e. it must be fulfilled as soon as the associated context is true and is no more active when the context becomes false. Generally some obligations might be enforced after some delay, typically if the subject of the obligation is a human operator. Those obligations are called obligations with deadlines [GF05, CCBB+08]. Enforcing obligations with deadline is more complex than immediate obligations. In order to simplify both the implementation and expression of obligations, we consider only immediate obligation in the remainder of this paper.

Other examples of reaction policies which demonstrate the need for permissions, obligations and prohibitions as part of the reaction policy are presented in section 6.

5.4 Reaction Process

The reaction process when an alert is received is the following:

1. creation of a threat organization org_{threat} associated with the new alert. org_{threat} is created as a sub-organization of the *supervision* organization.
2. activation of threat contexts in org_{threat}. Yet no concrete rules are inferred since no concrete entities are assigned to abstract entities in org_{threat}.
3. creation of concrete entities from the alert mapping. Role, activity and view definitions are evaluated to extract the data necessary to create the concrete entities from the alert.
4. assignment of subjects, actions and objects created from the alert to intrusion roles, activities and views in org_{threat}. This step and the previous one are specified through the abstract entity definitions.
5. activation of abstract security rules associated with the threat context into org_{threat}.
6. derivation of concrete security rules from activated abstract rules.
7. deployment of concrete security rules to configure security components.

The last step in this process is not covered by this article but an approach to deploy dynamic contextual security policies is proposed in [CCBB+08].

6 Reaction Policy Examples

In this section we present three examples of reaction policy. The first example specifies how to react when a buffer overflow is detected. The second and third examples demonstrate how to react to multi-step attacks: a brute force attack and a distributed denial of service (DDOS). The multi-step attacks examples assume that the information system on which the attack is detected uses a correlation engine to correlate the elementary attacks generated during the execution of the multi-step attacks (such as [CAB+06]). This correlation engine is part of the supervision loop presented in section 2. The reaction policy instantiation process is detailed in section 6.2.

6.1 Reacting against a Buffer Overflow

A possible counter-measure against a buffer overflow (BOF) is to isolate the machine from the network to correct the exploited vulnerability. We take the example of a BOF vulnerability in the WebDAV-enabled IIS 5.0 servers from Microsoft. The default threat abstract entities (see section 5) are used to write the abstract rules. Additionally, we define the *all_protocol* activity which abstracts the network protocols used by the machines involved in the attack, the *backup_web_server* view which abstracts the web server(s) started to replace a stopped web server and the *to_any_address* view which abstracts any ip address. *webdav_bof_ctx* is the threat context defined for this attack. We define the following obligations and prohibitions:

- the victim must be isolated from the rest of the network, i.e. all traffic is prohibited from and to the victim.:
  $prohibition(supervision, victim, all\_protocol, to\_any\_address, webdav\_bof\_ctx)$
  $prohibition(supervision, any\_address, all\_protocol, to\_victim, webdav\_bof\_ctx)$
- obligation to start a backup web server while the attacked server is unavailable:
  $obligation(supervision, administrator, start, backup\_web\_server, webdav\_bof\_ctx)$
- the vulnerable web server must be patched to remove the vulnerability:
  $obligation(supervision, administrator, update, to\_victim, webdav\_bof\_ctx)$

Notice that, in that reaction scenario, all reaction rules may be activated in parallel.

6.2 Reacting against Multi-step Attacks

Brute force attack. An example of attack requiring an alert correlation engine is a brute force attack. As said at the beginning of section 6, we assume an alert correlation engine has generated a global alert by fusing the elementary alerts corresponding to several failed logins on the same account using *ssh*.

As for the previous example, we introduce some additional abstract entities to express the reaction policy. The *rdp* role represents the components which deploy the reaction policy. The *victim_user* role abstracts the user being targeted by the attack. The *send_reset* activity abstracts the action(s) that disconnect the attacker from the victim. The *suspend_account* activity abstracts the action(s) taken by the administrator or a process to suspend a user account (which can be implemented by a script for different OSs like Linux/Unix, Microsoft windows, etc...). The *change_password* activity abstracts the actions that implement a password change. The *send_warning* activity abstracts the way a message is sent to a user to warn him/her that his/her account has been compromised. Finally we define a *to_victim_account* view which abstracts the users accounts. The mapping between an IDMEF alert and the *victim_user* role and *to_victim_account* view is done through the following role and view definitions (cf section 5.2):

$$empower(Threat\_org, Subject, victim\_user) :- threat\_context\_management(Alert, Threat\_org), mapping\_target\_user\_name(Alert, Subject).$$
$$use(Threat\_org, Object, to\_victim\_account) :- threat\_context\_management(Alert, Threat\_org), mapping\_target\_user(Alert, Object).$$

The following rules correspond to the RR example given in section 2 ($brute\_force\_ctx$ is the threat context defined for this attack):

- all traffic coming from the attacker and going to the victim is prohibited:
  $prohibition(supervision, attacker, all\_protocol, to\_victim, brute\_force\_ctx)$
- the connexion between the attacker and victim must be interrupted:
  $obligation(supervision, rdp, send\_reset, to\_attacker, brute\_force\_ctx)$

- the victim account must be suspended:
  obligation(supervision, rdp, suspend_account, to_victim_account, brute_force_ctx)
- the victim must be warned that his/her account has been attacked:
  $obligation(supervision, rdp, send\_warning, to\_victim\_user, brute\_force\_ctx)$
- once he/she has been warned that his/her account has been attacked, the victim must change his/her password:
  $obligation(supervision, victim\_user, change\_password, to\_victim\_account, brute\_force\_ctx \ \& \ received\_warning\_ctx)$
  The $received\_warning\_ctx$ context is true if the subject has received a warning through an email for example. Note the use of the context conjunction operator &.

Reaction policy instantiation example. Consider a brute force attack performed by a machine having ip address ip_A on a computer having for ip address ip_V against the account ac_V of user $user_V$. Suppose that an Intrusion Detection System (IDS) detects the attack and generates an alert m. The reaction policy for this attack occurrence is instantiated as follows:

1. When alert m is received by the reaction module, a threat organization $threat\_org_1$ is created as a sub-organization of organization *supervision*.
2. The threat context associated with the alert classification of m, namely the $brute\_force\_ctx$ context, is activated.
3. The relevant role and view definitions are evaluated so that the following statements become true:
   $empower(threat\_org_1, ip_A, attacker)$, $empower(threat\_org_1, ip_V, victim)$,
   $empower(threat\_org_1, user_V, victim\_user)$,
   $use(threat\_org_1, ac_V, to\_victim\_account)$.
   On the other hand we suppose the following statements are already true before the attack takes place and have been introduced in the policy by an administrator:
   - $empower(supervision, rdp\_host, rdp)$
   - $consider(supervision, tcp\_reset, send\_reset)$
   - $consider(supervision, tcp, all\_protocol)$
   - $consider(supervision, udp, all\_protocol)$
   - $consider(supervision, suspendacct, suspend\_account)$
   - $consider(supervision, send\_warning\_email, send\_warning)$
   - $consider(supervision, passwd, change\_password)$
4. Since the $brute\_force\_ctx$ context is true and given the aforementioned assignments of concrete entities to abstract entities, the following concrete security rules are derived:
   - $is\_obliged(rdp\_host, tcp\_reset, ip_A)$
   - $is\_prohibited(ip_A, tcp, ip_V)$
   - $is\_obliged(rdp\_host, suspendacct, ac_V)$
   - $is\_obliged(rdp\_host, send\_warning\_email, user_V)$

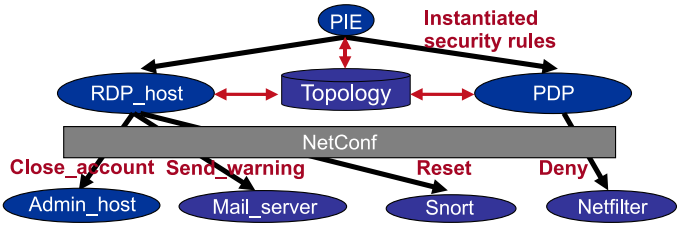


Fig. 2. The deployment architecture. The netconf [NWGN] protocol is used to configure some components of the information system. The PIE (Policy Instantiation Engine) is responsible for managing the global security policy. A PDP (Policy Decision Point) dispatches and translates concrete permissions and prohibitions. The RDP_host is responsible for the deployment of obligations.

The *suspendacct* is the name of a script which can be used under POSIX compatible OSs to suspend a user account. Note that when the user receives the warning, the $received\_warning\_ctx$ is activated, thus the conjunction $brute\_force\_ctx$ & $received\_warning\_ctx$ becomes true and the following obligation is inferred:

- $is\_obliged(user_V, passwd, ac_V)$

  Here *passwd* is the command line tool used to change a user password.

5. The concrete inferred security rules are deployed (figure 2). Please refer to [CCBB+08] for an example of reaction policy deployment architecture.

Distributed denial of service. We take the example of a DDOS attack implemented with the trinoo attack tool [Dit99]. An attacker uses the trinoo tool by connecting his/her computer to computers running the trinoo master program. Those master computers send commands to slave computers which generate attack traffic to specified target computers. The slave computers run the trinoo slave program. As in the previous example, we assume an alert correlation engine has detected the full DDOS scenario, hence an alert containing all the information about the master(s), slave(s) and target(s) of the attack has been generated.

Since the trinoo attack involves master and slave computers, we refine the generic abstract entity *attacker* by creating two sub-roles *master* and *slave*. We can still use the *attacker* role if for example the same rules must be applied to the master(s) and slave(s) machines (section 4, hierarchies). Figure 3 shows the ports and protocols used by the tool.

The number of machines composing the attack network may vary a lot. Hence formalizing the reaction policy as an abstract OrBAC policy is interesting as the number of abstract rules is independent from the attack network size. Moreover a trinoo network is composed of different types of attackers: the main attacker who controls the trinoo network, the master machine(s) and the slave machine(s). We can abstract all those types of attackers as different roles and manage them into the same dynamic threat organization. We define the following rules:

Fig. 3. Ports and protocols used by Trinoo

- master and slave machine(s):
  $prohibition(supervision, attacker, all\_protocols, to\_any\_address, trinoo\_ddos\_ctx)$
  $prohibition(supervision, any\_address, all\_protocols, to\_attacker, trinoo\_ddos\_ctx)$
  All traffic is prohibited from and to the master(s) and slave(s). Enforcing the concrete rules derived from those abstract rules may be impossible if the computers are outside of the information system. In such a case it is still possible to filter the incoming traffic. Note that since we want to block the outgoing and incoming traffic of the master(s) and slave(s), we use the *attacker* role and the *to_attacker* view in the rule so that the rules are inherited through the hierarchies.
- master machine(s):
  The master machine(s) can possibly be inside the information system monitored by the IDSs (a malicious employee can install them), so an alternative way of disabling them is to kill the process making them master(s):
  $$obligation(supervision, administrator, kill\_master\_process, to\_master, trinoo\_ddos\_ctx \ \& \ is\_ip)$$
- slave machine(s):
  The same applies for the slave(s):
  $$obligation(supervision, administrator, kill\_slave\_process, to\_slave, trinoo\_ddos\_ctx \ \& \ is\_ip)$$
- victim machine(s):

  $prohibition(supervision, slave, udp\_protocol, to\_victim, trinoo\_ddos\_ctx)$

  UDP traffic between the slave(s) and victim(s) is prohibited.

The *is_ip* context is true when a given ip address is part of the information system network:

$$hold(Org, \_, \_, O, is\_ip) : -ip\_belong\_to\_information\_system(O)$$

Note that we use a context conjunction operator & in the two obligations so that those two rules apply only to computers inside the information system. The following role definitions are used to map the alert onto the *slave* and *master* sub-roles:

$empower(Threat\_org, Subject, master) : -$
$threat\_context\_management(Alert, Threat\_org)$,
$mapping\_source\_master(Alert, Subject)$.

$empower(Threat\_org, Subject, slave) : -$
$threat\_context\_management(Alert, Threat\_org)$,
$mapping\_source\_slave(Alert, Subject)$.

7 Implementation

Our approach is implemented as part of the supervision platform presented in section 2.

The MotOrBAC support tool [ACCBC08] is used to specify the reaction policy (as well as the entire information system policy). The user can write the abstract reaction policy and use the simulation window to test his/her policy. Actually the user can load IDMEF alerts in the simulation window and see the concrete security rules inferred by MotOrBAC. MotOrBAC is able to assist the user during steps 1 to 6 in the list presented in section 5.4. The 7^{th} step consists in translating the concrete security rules inferred from the abstract policy into a set of languages used by the components implementing the security mechanisms as illustrated in figure 2 and further explained in [CCBB+08].

Figure 4 shows the abstract obligations defined with MotOrBAC to specify the reaction policy for the brute force attack example. One can see that this list of rules contains the rules presented in section 6.2 to react against a brute force attack. The same figure also shows an example of concrete security rules inferred when a new alert is received.

Abstract entity definitions for dynamic organizations are specified differently from other entity definitions because the organization in which the definition is given does not exist at the time of specification. Instead of selecting the organization in which the abstract entity definition is specified, the user chooses the type *Threat organization* and specify explicitly the mapping between the IDMEF alerts and the roles, activities and views using XPath expressions. The IDMEF alert generated by the correlation engine for the trinoo scenario can contain multiple instances of the *Source* IDMEF class. In section 6.2, the proposed reaction policy is different for the slave and master computers, hence their corresponding *Source* class instances in the IDMEF alert must be identified. The correlation engine segregates the master and slave computers in the alert by using the IDMEF *Process* class of the *Source* class. The *name* attribute of the *Process* class is used to hold the names segregating the master computer(s) (*master* process) and slave computer(s) (*ns* process).

Rule name	Organization	Role	Activity	View	Context
oblig3	supervision	rdp	send_warning	to_victim_user	brute_force
oblig4	supervision	victim_user	change_password	to_victim_account	composed_ctx
oblig1	supervision	rdp	send_reset	to_attacker	brute_force
oblig2	supervision	rdp	suspend_account	to_victim_account	brute_force

Type	Derives from	Subject	Action	Object
prohibition	prohib1	IP10_0_0_50	udp	IP10_0_0_60
prohibition	prohib1	IP10_0_0_50	tcp	IP10_0_0_60
obligation	oblig2	rdp_host	suspendacct	victim_user
obligation	oblig3	rdp_host	email_warning	victim_user
obligation	oblig1	rdp_host	tcp_reset	IP10_0_0_50
obligation	oblig4	victim_user	passwd	victim_user

Fig. 4. From top to bottom: the list of abstract obligations defined in motorbac for the brute force attack example and the concrete security rules derived from a sample IDMEF alert. The prohibitions are in red and the obligations in blue. The last obligation is not activated since the conjunction of contexts specified in the abstract rule from which it derives is false.

8 Conclusion

Managing the security of an information system involves defining a security policy and enforcing it. The supervision loop introduced in section 2 mentions the role of intrusion detection and the processing of its results, which involves reacting against attackers. The way the security components behave when an attack is detected can be formalized as a reaction policy. In this article, we showed that the OrBAC model is sufficiently expressive to model such policies. Although the RBAC model lacks expressiveness to specify such type of policy, we have shown that the OrBAC concept of dynamic organization is well suited to manage the activation of security rules upon the detection of an attack. Threat contexts can be combined with other types of contexts to express complex conditions. Since the reaction policy is expressed in the OrBAC model, it can be analyzed so that conflicts between rules can be detected and solved.

We have defined default intrusion roles, activities and views to manage most intrusions which involve one attacker and one victim. For more complex attacks with multiple attackers and victims, such as a DDOS, the default abstract entities can be refined. We have implemented the approach in the OrBAC API library, as a result the MotOrBAC support tool can be used to edit a reaction policy and help the policy administrators analyze and simulate it. We have integrated the OrBAC API in the implementation of a PIE to enable the automatic instantiation of reaction policies.

Some aspects have not yet been covered in this article, such as the lifetime of dynamic organizations, i.e. the lifetime of the concrete security rules deployed for each detected attack.

Acknowledgements. This work is partially funded by the Polux project (ANR 06-SETIN-012).

References

[ACCBC08] Autrel, F., Cuppens, F., Cuppens-Boulahia, N., Coma, C.: Motorbac 2: a security policy tool. In: Third Joint Conference on Security in Networks Architectures and Security of Information Systems (SARSSI) (2008)

[Bra98] Brackney, R.: Cyber-intrusion response. In: Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems (1998)

[CAB+06] Cuppens, F., Autrel, F., Bouzida, Y., Garcia, J., Gombault, S., Sans, T.: Anti-correlation as a criterion to select appropriate counter-measures in an intrusion detection framework (2006)

[CBCdV+08] Cuppens-Boulahia, N., Cuppens, F., Lopez de Vergara, J.E., Vazquez, E., Guerra, J., Debar, H.: An ontology-based approach to react to network attacks. In: Third International Conference on Risk and Security of Internet and Systems (CRiSIS 2008) (2008)

[CCB08] Cuppens, F., Cuppens-Boulahia, N.: Modeling contextual security policies. International Journal of Information Security (IJIS) 7(4) (August 2008)

[CCBB+08] Cuppens, F., Cuppens-Boulahia, N., Bouzida, Y., Kanoun, W., Croissant, A.: Expression and deployment of reaction policies. In: SITIS Workshop Web-Based Information Technologies and Distributed Systems (WITDS), Bali, Indonesia (2008)

[CCBG07] Cuppens, F., Cuppens-Boulahia, N., Ben Ghorbel, M.: High-level conflict management strategies in advanced access control models. Electronic Notes in Theoretical Computer Science (ENTCS) (2007)

[CCBM04] Cuppens, F., Cuppens-Boulahia, N., Miège, A.: Inheritance hierarchies in the Or-BAC model and application in a network environment. In: Second Foundations of Computer Security Workshop (FCS 2004) (2004)

[CP00] Carver, C.A., Pooch, U.W.: An intrusion response taxonomy and its role in automatic intrusion response. In: IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop (2000)

[Dit99] Dittrich, D.: The DoS project's trinoo distributed denial of service attack tool (1999), `http://staff.washington.edu/dittrich/misc/trinoo.analysis`

[DTCCB07] Debar, H., Thomas, Y., Cuppens, F., Cuppens-Boulahia, N.: Enabling automated threat response through the use of a dynamic security policy. Journal in Computer Virology 3(3) (2007)

[Fis96] Fisch, E.A.: A taxonomy and implementation of automated responses to intrusive behavior. PhD thesis, Texas A and M University (1996)

[FSG+01] Ferrailo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST standard for rbac. ACM Transactions on Information and System Security (2001)

[GF05] Gama, P., Ferreira, P.: Obligation policies: An enforcement platform. In: IEEE International Workshop on Policies for Distributed Systems and Networks (2005)

[HCF07] Debar, H., Curry, D., Feinstein, B.: The intrusion detection message exchange format (idmef) (2007)

[KBB+03] Abou El Kalam, A., El Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarteand, Y., Miège, A., Saurel, C., Trouessin, G.: Organization based access control. In: IEEE 4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003) (2003)

[MF03] Motta, G.H.M.B., Furuie, S.S.: A contextual role-based access control authorization model for electronic patient record. IEEE Transactions on information technology in biomedicine 7(3) (2003)

[NWGN] NETCONF Working Group. Netconf., `http://tools.ietf.org/wg/netconf/trac/wiki`

[PCBC+07] Preda, S., Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J., Toutain, L.: Reliable process for security policy deployment. In: International Conference on Security and Cryptography (Secrypt 2007) (2007)

[SBW07] Stakhanova, N., Basu, S., Wong, J.: A taxonomy of intrusion response systems. International Journal of Information and Computer Security 1(1/2), 169–184 (2007)

[Ull89] Ullman, J.D.: Principles of database and knowledge-base systems. Computer Science Press (1989)

Towards System Integrity Protection with Graph-Based Policy Analysis

Wenjuan Xu[1], Xinwen Zhang[2], and Gail-Joon Ahn[3]

[1] University of North Carolina at Charlotte
wxu2@uncc.edu
[2] Samsung Information Systems America
xinwen.z@samsung.com
[3] Arizona State University
gahn@asu.edu

Abstract. Identifying and protecting the trusted computing base (TCB) of a system is an important task, which is typically performed by designing and enforcing a system security policy and verifying whether an existing policy satisfies security objectives. To efficiently support these, an intuitive and cognitive policy analysis mechanism is desired for policy designers or security administrators due to the high complexity of policy configurations in contemporary systems. In this paper, we present a graph-based policy analysis methodology to identify TCBs with the consideration of different system applications and services. Through identifying information flows violating the integrity protection of TCBs, we also propose resolving principles to using our developed graph-based policy analysis tool.

1 Introduction

In an operating system, an information flow occurs when one process can write to a resource (e.g., device or file) which can be read by another process. Inter-process communication between these processes may also cause possible information flows. Integrity goal is violated if there exists information flow from an unprivileged process to a privileged process or between two different applications or services where information flow should be controlled. In a typical system, information flows are controlled through security policies. Our objective in this paper is to provide an effective framework for analyzing system security policies and finding policy rules causing information flows with integrity violations.

There are several related approaches and tools to analyze policies based on certain information flow models [1,9,11,16]. Most of these work focus on the identification of a common and minimum system TCB which includes trusted processes for an entire system. These approaches can analyze whether a policy meets security requirements such as no information flow from low integrity processes (e.g., unprivileged user processes) to system TCB (e.g., kernel and init). However, they cannot identify integrity violations for high level services and applications that do not belong to system TCB. In practical, other than system TCB protection, a high level application or system service is required to achieve integrity assurance through controlling any information flowing from other

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 65–80, 2009.

applications. An existing argument in [20] clearly states the situation as follows: "a network server process under a UNIX-like operating system might fall victim to a security breach and compromise an important part of the system's security, yet is not part of the operating system's TCB." Accordingly, a more comprehensive policy analysis for TCB identification and integrity violations is desired.

Another issue in policy analysis is the large size and high complexity of typical policies in contemporary systems. For example, an SELinux [12] policy has over 30,000 policy statements in a desktop environment. Under this situation, several challenges exist for system designers or administrators in the context of policy analysis. We enumerate such challenges to derive the motivation of this paper: (1) *Understanding and querying a policy* All previous policy query tools lack an effective mechanism for a user to understand a particular policy. Furthermore, without understanding a policy, the user even cannot figure out what to query; (2) *Recognizing information flow paths* In the work of information flow based policy analysis [9,16], information flow paths are expressed with text-based expressions. However, primitive text-based explanations cannot provide appropriate degree of clarity and visibility for describing flow paths; and (3) *Identifying integrity violation patterns, inducements, and aftermaths* In the work of Jaeger et al. [11,17], they elaborate violation patterns using graphs. However, their approach does not leverage the features and properties of graphs for analyzing policies and identifying the causes and effects of integrity violations.

In this paper, we consider an information domain as a collection of *subjects* (e.g., processes) and *objects* (e.g., files, ports, devs) which jointly function for an application or service. We further define the *domain TCB* as subjects which should have the same integrity level in the domain. The integrity of the domain cannot be judged unless information flows between this domain TCB and the rest of the system are appropriately controlled. Based on these notions, we propose a *domain-based integrity model*. Based on this model, we build a graph-based policy analysis methodology for identifying policy rules that cause integrity violations (or simply *policy violations*). Our graph-based approach can provide several benefits since information visualization helps a user heuristically explore, query, analyze, reason, and explain obtained information. Also, our graph-assisted approach simplifies analysis and verification tasks while rigorously producing distinctive representation of analysis results. In addition, we describe a set of principles for resolving identified policy violations in policy graphs. A graph-based policy analysis tool (GPA) is also developed based on our approach.

The rest of this paper is organized as follows. Section 2 describes background and some work related to our graph-based policy analysis. The principles and methodology of our work are illustrated in Section 3. Section 4 presents how to develop and use GPA for analyzing policies. In this section, we adopt SELinux policies as an example. Section 5 concludes this paper presents our future work.

2 Background and Related Work

2.1 Trusted Computing Base

The concept of TCB partitions a system into two parts: the part inside TCB which is referred as to be trusted (TCB) and the part outside TCB which is referred as to be

untrusted (NON-TCB). Therefore, the identification of TCB is always a basic problem in security policy design and management. The famous Orange Book [2] proposes TCB as part of a system that is responsible for enforcing information security policies. Reference monitor-based approach is proposed in [4], where a system's TCB not only includes reference monitor components, but also encompasses all other functionalities that directly or indirectly affect the correct operation of the reference monitor such as object managers and policy database. Considering an operating system, its TCB includes kernel, system utilities and daemons as well as all kinds of object management and access control functions. Typical object management functions are responsible for creating objects and processing requests while typical access control functions consist of both rules and security attributes that support decision-making for access control.

2.2 Integrity Model

To describe information flow-based integrity protection, various models are proposed and developed in past years, such as Biba [7], Clark-Wilson [15], LOMAC [19] and CW-lite [17]. Biba integrity property is fulfilled if a high integrity process cannot read lower integrity data, execute lower integrity programs, or obtain lower integrity data in any other manner. LOMAC supports high integrity process's reading low integrity data, while downgrading the process's integrity level to the lowest level that it has ever read. Clark-Wilson provides a different view of dependencies, which states that through certain programs so-called transaction procedures (TP), information can flow from low integrity objects to high integrity objects. Later the concept of TP is evolved into filter in CW-Lite model. A filter can be a firewall, an authentication process, or a program interface for downgrading or upgrading the privileges of a process. In CW-lite model, information can flow from low integrity processes (NON-TCB) to high integrity processes (TCB) through filters.

2.3 Policy Analysis

The closest existing work to ours include Jaeger et al. [11] and Shankar et al. [17]. In these work, they use a tool called Gokyo for checking the integrity of a proposed TCB for SELinux [18]. Also, they propose to implement their idea in an automatic way. Gokyo mainly identifies a common TCB in SELinux but a typical system may have multiple applications and services with variant trust relationships. Still, achieving the integrity assurance for these applications and services is not addressed in Gokyo.

Several query-based policy analysis tools have been developed. APOL [1] is a tool developed by Tresys Technology to analyze SELinux policies. SLAT (Security Enhanced Linux Analysis Tool) [9] defines an information flow model and policies are analyzed based on this model. PAL (Policy Analysis using Logic Programming) [16] uses SLAT information flow model to implement a framework for analyzing SELinux policies. All these tools try to provide a way for querying policies. However, they all display policies and policy query results in text-based expressions, which are difficult to understand for policy developers or security administrators. Other policy analysis methods are also proposed. For example, in [23], they propose to analyze the policies with information-flow based method. For another example, in [22], they try to analyze

the policies with graph-based model. However, non these approaches are applicable in our scenarios since they are specific to certain access control model, and none of them are realized with certain tools.

To overcome these issues, we have developed a graph-based methodology for identifying and expressing interested information flows in SELinux policies [21]. We also have proposed a policy analysis mechanism using Petri Nets is proposed in [3]. However, this work does not have the capability of policy query, thus is limited in identifying system TCB and other domain TCBs.

3 Graph-Based Policy Analysis

To help a policy administrator better understand security policies and perform policy analysis tasks, we first propose a graph-based policy analysis methodology in this section. Graphs leverage highly-developed human visual systems to achieve rapid uptake of abstract information [10]. In our methodology, we have two parallel building blocks: basic policy analysis and graph-based policy analysis. The basic policy analysis is composed of security policy definitions, integrity model for identifying policy violations, and methods for resolving policy violations. Graph-based analysis is built according to basic policy analysis and expresses corresponding components with graphical mechanisms and algorithms. We elaborate the details of our methodology in the remainder of this section.

3.1 Basic Policy Analysis

Security Policies. A security *policy* is composed of a set of subjects, a set of objects, and a set of *policy statements or rules* which states that a subject can perform what kind of actions on an object. For information flow purpose, all operations between subjects and objects can be classified as $write\_like$ or $read\_like$ [9] and operations between subjects can be expressed as $calls$. Depending on the types of operations, information flow relationships can be identified. If subject *x* can write to object *y*, then there is information flow from *x* to *y*, which is denoted as $write(x, y)$. On the other hand, if subject *x* can read object *y*, then there is information flow from *y* to *x* denoted as $read(y, x)$. Another situation is that if subject *x* can call another subject *y*, then there is information flow from *y* to *x*, which is denoted as $call(y, x)$. Moreover, the information flow relationships between subjects and objects can be further described through *flow transitions*. In a policy, if a subject s_1 can write to an object *o* which can be read by another subject s_2, then it implies that there is an *information flow transition* from subject s_1 to s_2, denoted as $flowtrans(s_1, s_2)$. Also, if subject s_2 can call a subject s_1, there is a flow transition from s_1 to s_2. A sequence of flow transitions between two subjects represents an *information flow path*.

Integrity Model. Retrospecting the integrity models introduced in Section 2, one-way information flow with Biba would not be sufficient for many cases as communication and collaboration between application or service domains are frequently required in most systems. Although filters between high and low integrity data are sufficient enough for TCB and NON-TCB isolations, it is not suitable for application or service domain

isolations. For example, processes of user applications and staff applications are required to be isolated since both are beyond the minimum and common TCB boundary. With that reason, we develop a *domain-based integrity model*, in which a concept called *domain TCB* is defined to describe subjects and objects required to be isolated for an information domain. To be clear, the minimum and common TCB of a system is called *system TCB* in our paper. Also, for a subject in a system, if it neither belongs to the system TCB, nor belongs to the domain TCB of a particular application or service, then it is in the NON-TCB of the system.

Information Domain. As mentioned in Section 1, an application or service information domain consists of a set of subjects and objects. Here, we propose two steps to identify an information domain.

- *Step1: Keyword-based domain identification.* Generally, subjects and objects in a security policy are described based on their functions, e.g., *http* is always the prefix for describing web server subjects and objects in SELinux. Hence, to identify the web server domain, we use keyword *http* to identify the initial set of subjects and objects in this domain.
- *Step2: Flow-based domain identification.* In a security policy, some subjects or objects cannot be identified through keyword prefix. However, they can flow to initially identified domain subjects and objects, influencing the integrity of this domain. Therefore, we also need to include these subjects and objects into the domain. For instance, in a Linux system, *var* files can be read by web server subjects such as *httpd*. Hence they should be included in the web server domain.

Domain TCB. To protect the integrity of an information domain, a domain TCB is defined. TCB(d) (domain *d*'s TCB) is composed of a set of subjects and objects in domain *d* which have the same level of security sensitivity. In other words, a web server domain running in a system consists of many subjects–such as processes, plugins, and tools, and other objects including data files, configuration files, and logs. We consider all of these subjects and objects as TCB of this domain, while its network object such as $tcp : 80$ (`http_port_t`) is not considered as TCB since it may accept low integrity data from low integrity subjects. In a system, the integrity of an object is determined by the integrity of subjects that have operations on this object. Hence, we need to identify TCB(d) subjects of each information domain and verify the assurance of their integrity.

To ease this task, a minimum TCB(d) is preferred. However, in the situation that the minimum TCB(d) subjects have dependency relationships with other subjects, these other subjects should be added to domain TCB or dependencies should be removed. Based on these principles, we first identify an initial TCB(d) subjects which are predominant subjects for domain *d*. We further discover TCB(d) considering subject dependency relationships with the initial TCB(d) through *flow transition-based identification* and *violation-based adjustment*.

- *Step1: Initial TCB(d) identification.* In an information domain, there always exist one or several predominant subjects, which launch all or most of other subjects functioning in this domain. Here, we identify the initial TCB(d) subjects based on

these subject launching relationships and the number of subjects that a subject can launch. For example, for web server domain, *httpd* launches all other processes like *httpd_script*, hence it belongs to the initial TCB(d) of this domain.
- *Step2: Flow transition-based TCB(d) identification.* The subjects that can flow only to and from the initial identified TCB(d) are included into domain TCB. For instance, if subject *httpd_php* can flow only to and from *httpd*, then *httpd_php* should be included into TCB(d).
- *Step3: TCB(d) adjustment by resolving policy violations.* After identifying policy violations (or integrity violations described shortly in this subsection), we adjust the identified TCB(d) with wrongly included or excluded subjects. For example, initially subject *awstats_script* (web server statistics script) is excluded from TCB(d). After identifying policy violations caused from *awstats_script* to web server TCB(d), we found that these violations can be ignored. Hence, the TCB(d) should be adjusted to include *awstats_script*.

Domain-based Integrity Model. Based on the concept of system TCB and TCB(d), a domain-based integrity model is defined as follows.

Definition 1. *Domain-based integrity model is satisfied for an information domain d if for any information flow to TCB(d), the information flow path is within TCB(d); or the information flow path is from the system TCB to TCB(d); or the information flow path is from another domain TCB and it is filtered.*

Through this definition, domain-based integrity model achieves the integrity of an information domain by isolating information flow to TCB(d). This model requires that any information flow happening in a domain *d* adheres within the TCB(d), from system TCB to the TCB(d), or from another domain TCB via filter(s). In this paper we do not discuss the integrity of filters, which can be ensured with other mechanisms such as formal verification or integrity measurement and attestation [14]. Filters can be processes or interfaces that normally is a distinct input information channel and is created by, e.g., a particular open(), accept() or other call that enables data input. For example, linux *su* process allows a low integrity process (e.g., staff) changes to be high integrity process (e.g., root) through calling *passwd* process. For another example, high integrity process (e.g., httpd administration) can accept low integrity information (e.g, network data) through the secure channel such as *sshd*. Normally, it is the developer's tasks to build filtering interfaces and prove effectiveness to the community [13]. Generally, without viewing system application codes, an easier way for policy administrator to identify filters is to declare filters with clear annotations during policy development [17]. Here, we assume that filters can be identified through annotations. Also, in our work, initially we do not have a set of predefined filters. After detecting a possible policy violation, we identify or introduce a filter subject to resolve policy violations.

Policy Violation Detection. Based on domain-based integrity model, we treat a TCB(d) as an isolated information domain. We propose the following rules for articulating possible policy violations for system TCB and TCB(d) protections.

Rule 1. *If there is information flow to a system TCB from its subjects without passing any filter, there is a policy violation for protecting the system TCB.*

Rule 2. *If there is information flow from TCB(d_x) to TCB(d_y) without passing any filter, there is a policy violation in protecting TCB(d_y).*

Policy Violation Resolution. After possible policy violations are identified with violation detection rules, we take systematic strategies to resolve them. Basically, for a violation, we first evaluate if it can be resolved by adding or removing related subjects to/from system or domain TCBs. This causes no change to the policy. Secondly, we try to identify if there is a filter along with the information flow path that causes the violation. If a filter can be identified, then the violation is a false alarm and there is no change to the policy graph. Thirdly, we attempt to modify policy, either by excluding subjects or objects from the violated information flow path, or by replacing subjects or objects with more restricted privileges. In addition, we can also introduce a filter subject that acts as a gateway between unauthorized subjects and protected subjects.

3.2 Graph-Based Analysis

Semantic substrates [5] is a visualization methodology for laying out a graph, in which graph nodes are displayed in non-overlapping regions based on node attributes. Through this way, the location of a node conveys its information. Also, the visibility of graphic links used to describe node relationships is available depending on user control. In our work, we use semantic substrates to display *policy graph* and *policy violation graph* which are defined based on the previously stated security policies and policy violations. A graphical query-based violation identification method is introduced based on domain-based integrity model. Also, we illustrate how we can apply policy violation resolution methods to policy violation graphs.

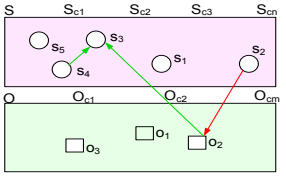
Policy Graph. A security policy consists of a set of subjects, objects, and operations including *write, read* and *call*. We define a policy graph as follows:

Definition 2. *A* Policy Graph *of a system is a directed graph G=(V, E), where the set of vertices V represents all subjects and objects in the system, and the set of edges E=$V \times V$ represents all information flow relations between subjects and objects. That is,*

- *V=$V_o \bigcup V_s$, where V_o and V_s are the sets of nodes that represent objects and subjects, respectively;*
- *E=$E_r \bigcup E_w \bigcup E_c$. Given the vertices v_{s1},$v_{s2} \in V_s$ separately representing subject s1 and s2, and vertices $v_o \in V_o$ representing object o, $(v_{s1}, v_o) \in E_w$ if and only if $write(s1, o)$, $(v_o, v_{s2}) \in E_r$ if and only if $read(o, s2)$, and $(v_{s1}, v_{s2}) \in E_c$ if and only if $call(s1, s2)$.*

As concluded in [8], humans perceive data coded in spatial dimensions far more easily than those coded in non-spatial ones. Based on this concept, we use semantic substrates to display policies. We divide a canvas into different areas based on the classification of entities (subjects and objects) and then layout nodes expressing the entities into corresponding areas. We also use non-spacial cues (e.g., color or shape) to emphasize certain nodes or a group of nodes. Figure 1 (a) shows the semantic substrates-based graph design. The Y-axis is divided into regions, where each region contains nodes representing

entities such as subjects and objects. Furthermore, in each region, nodes representing entities of different classifications are placed in different spaces along with the X-axis. For subjects and objects in a policy, $S_{c1}...S_{cn}$ and $O_{c1}...O_{cm}$ separately represent certain classifications. Different colors and shapes are used to distinguish the identification of different nodes. Circles and rectangles are used to represent subjects and objects, respectively. Relationships between subjects and objects are expressed with lines in different colors or shapes. For instance, the write operation between subject s_2 and object o_2 is expressed with a red link.



(a) Design 1- Policy Graph: Links between S_2 and O_2 represents write operation; between S_3 and O_2 expresses read operation; between S_4 and S_3 denotes call operation

Algorithm 1: [Building Policy Graph]
Input: The Policy file *Policy*, the Policy Explanation File F_e, the Permission Mapping File F_p, the Subject Classification File F_s, the Object Classification File F_o.
Output: A Semantic-based Policy Graph *G*
Method:
(1) *Policy_t: = policyParsing(Policy,F_e, F_p, F_s, F_o);/* parsing the policies files into subjects, objects and relationships , and mapping the classification into the parsed policies.*
(2) *G = drawCanvas (Policy_t); /* constructing the canvas for drawing the graphs and also dividing the graphs into different areas.*
(3) *G = drawNodes (G , Policy_t); /* reading the entities for the policies and drawing them in nodes into the classified areas based on the Policy_t structure.*
(4) *G = drawLines (Policy_t, G , n); /* drawing the link from the node to the other nodes and setting the attribute for the link.*

(b) Algorithm for building policy graph

Fig. 1. Designation and algorithm for expressing policies in graph

Different security policies have different formats and components. To give a uniform way for policy analysis, we need to preprocess a primitive policy. Figure 1 (b) summarizes the procedures of policy graph representation. First, a policy file *Policy* is parsed and mapped through a policy explanation file F_e and a permission mapping file F_p. F_e includes meta information such as the format of the policy and subject/object attributes in the policy. The policy format can be binary or text and is organized in certain order. The subjects are users or processes and objects are system resources such as files, data, port or labels specifying these resources. F_p states operations between subjects and objects that are mapped to *write()*, *read()*, or *call()*. For instance, if a subject has an operation to get the attribute of an object, the operation is mapped to *read()*. In addition, F_s and F_o files separately define subject and object classifications in the system. After parsing the policy, a canvas is drawn and divided into different areas, on which nodes representing policy entities are drawn and relationships between them are expressed with arrows. Also, during the execution of this algorithm, policy rules are stored as attributes for corresponding graph nodes and arrows.

Violation Identification with Graphical Queries. According to our domain-based integrity model, we need to investigate information domain, domain TCB, and then identify policy violations. To complete these tasks, we design a graphical user interface to create and run queries against a policy graph, and then get required information such as which subject should be included in the information domain. A query formulation is built with four basic components–*Subject*, *Object*, *Direct flow* and *Indirect flow*. Figure 2 summarizes these visual components and details are elaborated as follows.

– *Subject* is shaped as a labelled circle node to indicate policy subject(s) in a query. A user can define the Subject node as a single subject or a set of subjects based

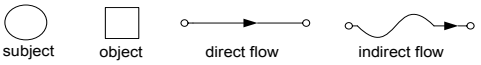


Fig. 2. Policy query building blocks

on different query requirements. The label under the Subject node specifies user defined subject names. For instance, a user can query to find a policy subject *httpd*, a set of policy subjects *httpd, php*, and *ssh*, or any sequence of policy subjects "*" from a policy graph. Color filled in the Subject node represents user defined policy subject attribute for a query. An example of the attribute can be the name of an information domain which is expressed in green color. Therefore, a user can create an example query with green color in the Subject node which is labelled with *httpd*. The meaning of this query is to find a subject which belongs to the information domain and has name *httpd*. After the subject is found, it is changed to an expected color (e.g., red), which is defined by the user through specifying the Subject node line color. Also, if the query purpose is to identify policy subject(s), then wildcard "?" should be marked on the Subject node.

- *Object* is represented as a labelled rectangle node to denote policy object(s) in a query. Similarly, the label under the Object node illustrates a single policy object (e.g., *etc*) or a set of policy object (e.g.,*etc* or *bin*). Also, a user can specify policy object attribute (e.g., domain name) for a query and the color of identified policy object in the result graph through coloring the Object node rectangle and rectangle line, respectively. In the situation that the query purpose is to identify objects, "?" is specified on the Object node.
- *Direct flow* is drawn with a link to connect Subject node and Object node and provides a way for direct information flow-based query. The label of a link represents the intended information flow. For example, a user can query to find if a write operation exists between a policy subject and a policy object. To specify that the intended query is to identify a direct flow, a user can denote the Direct flow link with "?".
- *Indirect flow* is expressed in a curved link to connect Subject node and Object node. The main purpose of indirect flow is to specify the intended information flow paths between subjects and objects. A user can find the shortest path, all paths, or any path. The wildcard "*" denotes all paths that can be found between subjects and objects. If the intended query is to identify indirect flow paths, "?" should be drawn on the Indirect flow link.

Information Domain Queries. Corresponding to two steps of information domain identification presented in Section 3.1, we propose two principles as follows.

- *Name-based policy subjects or objects query* To identify domain subjects and objects based on keyword, we construct subject and object queries by using their names. Figure 3 (a) shows an example query: *Identifying the subjects or objects whose names have prefix "prefix", and painting result nodes with green color*. The query result is shown in Figure 3 (a'), where subjects S_1, S_3 and object O_1 are identified to be included in the information domain.

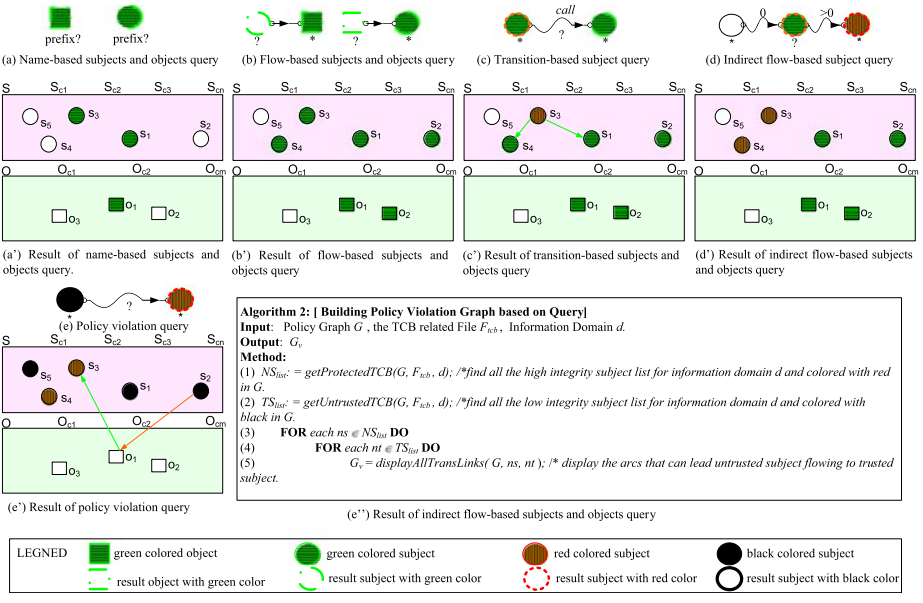


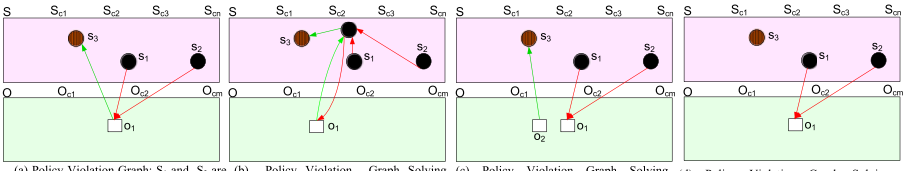
Fig. 3. Policy query examples

– *Direct flow-based subjects or objects query* To investigate domain subjects and objects based on direct flows, we construct an example query shown in Figure 3 (b). The meaning of this query is: *Finding the subjects or objects that can directly flow to the initial identified domain subjects and objects (green colored), and painting result nodes with green color*. The query result in Figure 3 (b') indicates that subjects S_2, S_4 and object O_2 should be added into the information domain.

Domain TCB Queries. For domain TCB identification steps described in Section 3.1, we query TCB(d) for an information domain with following principles. After TCB(d) queries are completed, TCB(d) subject information is saved into a file F_{tcb}.

– *Transition-based subjects query* To query a TCB(d), we first identify TCB(d) based on subject launching relationships. The example query in Figure 3 (c) states: *Identifying and displaying the direct call flow links between domain subjects*. Example result is shown in Figure 3 (c'), which displays the call transition relationships between subjects S_1, S_2 and S_3. Hence, subject S_3 belongs to TCB(d).
– *Indirect flow-based subjects query* To identify the subjects that can flow only to initial TCB(d) as shown in Figure 3 (d), a query is constructed as follows: *Identifying the subjects that belong to the information domain (green colored) can flow only to the initial TCB(d) (red colored) with red color in result nodes*. Figure 3 (d') indicates the example result that S_4 should be added into TCB(d).

Policy Violation Queries. Before introducing our methodology for policy violation queries, we first define a violation policy graph based on our proposed integrity model.



(a) Policy Violation Graph: S_1 and S_2 are NON-TCB subject which can flow to TCB or TCB(d) subject S_3 through object O_1

(b) Policy Violation Graph Solving through Filter, which can be transited by NON-TCB and TCB subjects and can read, write object O_1

(c) Policy Violation Graph Solving through adding new object O_2 which can be read by subject s_3

(d) Policy Violation Graph Solving through removing read operation between TCB or TCB(d) subject S_3 and O_1

Fig. 4. Policy violation and modification example graphs

Definition 3. *Given a policy graph $G = (V, E)$, the subject vertices belonging to NON-TCB, system TCB, and TCB(d) are represented by V_{NTCB}, V_{TCB}, and V_{TCBd}, respectively. A violation policy graph $G^v = (V^v, E^v)$ for domain d is a subgraph of G where*

- $V^v = \{v : v \in V_{NTCB}, \exists u : u \in V_{TCB} \cup V_{TCBd} \wedge (v, u) \in E\}$
- $E^v = \{(u, v) : u, v \in V^v \wedge (u, v) \in E\}$

Figure 3 (e') shows a violation path with a flow transition from a low integrity subject (black) to a high integrity subject (red). To generate the policy violation graph, a query is constructed as shown in Figure 3 (e), where we draw in the Subject node with black color (NON-TCB) and red color (TCB), trying to identify the policy violations from NON-TCB to TCB caused by indirect information flow. Figure 3 (e") shows the details of policy violation graph generation based on query. Through the query operations performed earlier, NON-TCB, TCB and TCB(d) are elaborated in a file F_{tcb}. Also, NON-TCB, TCB and TCB(d) subject nodes are separately colored. Then we discover all flow transitions from NON-TCB subjects to system TCB subjects or TCB(d) subjects. Note that it is optional for a policy administrator to specify queries from NON-TCB to TCBs through specifying the exact subject names rather than using "*".

Policy Violation Resolutions in Graph. With a generated policy violation graph, we introduce different approaches to modify the policy graph and remove policy violations and illustrate the expected graph result after the modification. Based on the policy violation resolution strategies discussed in Section 3.1, other than ignoring a policy violation through adding related subjects to system or domain TCBs, we can remove the violation by importing a filter subject. Comparing Figure 4 (a) with violation resolved graph in Figure 4 (b), write and read operations by the NON-TCB and TCB are removed, transition relationships between subjects and the filter are added, and the policy violations caused by NON-TCB subjects S_1 and S_2 are resolved. Another optional way for resolving policy violations is to import new subjects or objects to restrict original subjects or objects privileges. As shown in Figure 4 (c), new object O_2 is introduced so the information flows between NON-TCB and TCB are removed. Also, we can resolve the policy violations through deleting related policy statements. Example result of the modification is shown in Figure 4 (d), where the read operation between object O_1 and TCB subject S_3 is removed to resolve policy violations between NON-TCB and TCB.

4 Graph-Based Policy Analysis

Our previous policy visualization tool [21] has shown the feasibility and benefits to visualize security policies with *semantic substrates* [5]. Extended from this, a more comprehensive policy analysis tool (GPA) is developed in our work, which implements the query functionality for achieving high system assurance with information flow control. In this section we use an example SELinux reference policy to show the flexibility and efficiency of policy analysis with our tool. We use JDK1.6 and other necessary Java libraries to develop the main analysis components. We implemented graph drawing with graph package Piccolo [6].

Applying the reference monitor-based TCB identification method to SELinux, subjects functioning as reference monitor such as checking policy and loading policy belong to system TCB. Also, processes used to support reference monitor such as kernel and init, are included into system TCB. After reference monitor-based system TCB identification is completed, other subject types such as *restorecon* are required to be included into system TCB based on their flow transition relationship with the initial system TCB. Table 1 shows the TCB domains that we have recognized.

As an example, we use Apache web server as a target service domain to achieve high integrity. We first identify subjects and objects belonging to Apache domain. We then specify Apache TCB(d), list the policy violations identified against our integrity model, and resolve them with different principles.

Apache Domain Identification. To identify subjects and objects for Apache domain, we first apply keyword-based identification principle. As discussed earlier, we use *http* as a keyword prefix. As a result, subjects and objects such as `httpd_t`, `httpd_php_t` are included into Apache domain. With the flow-based identification principle, we discover all subjects and objects that have a direct flow to the initially identified Apache subjects and objects and include them into Apache domain. Table 1 shows a selected list of subjects and objects that we detected. Also, we use graph-based query functions implemented in GPA to automatically identify Apache information domain. Figures 5.I (a) and 5.I (b) show how we use graph queries to identify subjects and objects corresponding to Apache domain identification steps.

Apache TCB(d) Identification. Based on the initial TCB(d) identification principle in Section 3, we get initial TCB(d) subjects from Apache information domain. Specifically, our analysis shows that all subject domains in Apache related policy rules include a set of domain relationships since a domain `httpd_t` can transit to other `httpd` domains such as `httpd_php_t` and so on. Thus, a subject labelled by `httpd_t` is a predominant subject which launches other subjects in Apache server. Similarly, a subject labelled as `httpd_suexec_t` is also a predominant subject since this domain can transit to most of other `httpd` domains. Naturally, `httpd_t` and `httpd_suexec_t` are included into Apache TCB(d). Secondly, we construct a query to find all subjects that can transit only to the initially identified TCB(d) (shown in Figure 5.I (d)). Based on the generated query results, `httpd_sysadm_script_t`, `httpd_rotatelogs_t` and `httpd_php_t` can transit only to `httpd_t` and `httpd_suexec_t` other than system TCB subjects.

Violations and Resolutions for Apache. Based on the example query drawn in Figure 5.I (e), we automatically identify possible policy violations from NON-TCB subjects to TCB(d) subjects in Apache service. Figure 5(II) shows the identified policy violations, which implies that TCB(d) integrity is violated because NON-TCB subjects have write_like operations on objects that can be read by TCB(d) subjects. Due to possible high number of violations, our GPA tool can selectively show certain violations, allowing a policy administrator to solve them based on the priority.

Adjust TCB(d). After policy violations are identified, Apache TCB(d) is required to be adjusted and policy violations should be removed. As shown in Table 2, `httpd_awstat s_script_t` can flow to TCB(d) subjects through `httpd_awstats_script_rw_t`. At the same time, it is flown in by many NON-TCB subjects through some common types such as `devtty_t`. Hence, we ignore the violations caused by this *awstats_script* and include it into TCB(d). Similar situation occurs for `httpd_apcupsd_cgi_script_t` and `httpd_prewikka_script_t`. However,

Table 1. Apache information domain

System TCB				
kernel_t	load_policy_t	initrc_t	bootloader_t	quota_t
mount_t	ipsec_mgmt_t	useradd_t	automount_t	passwd_t
hwclock_t	admin_passwd_exec_t	cardmgr_t	checkpolicy_t	fsadm_t
kudzu_t	sshd_login_t	restorecon_t	newrole_t	klogd_t
syslogd_t	sysadm_t	getty_t	apt_t	sshd_t
dpkg_t	logrotate_t	snmpd_t	ldconfig_t	init_t
lvm_t	local_login_t	setfiles_t		
Identified Key word-based Apache Subjects and Objects				
Apache Subjects				
httpd_staff_script_t	httpd_awstats_script_t	httpd_t	httpd_rotatelogs_t	httpd_helper_t
httpd_unconfined_script_t	httpd_php_t	httpd_sysadm_script_t	httpd_sys_script_t	httpd_suexec_t
httpd_prewikka_script_t	httpd_apcupsd_cgi_script_t	httpd_user_script_t		
Apache Objects				
httpd_staff_script_ra_t	httpd_unconfined_script_ro_t	httpd_cache_t	httpd_user_script_rw_t	
httpd_user_script_exec_t	httpd_prewikka_script_ro_t	httpd_exec_t	httpd_apcupsd_cgi_script_ra_t	
httpd_user_htaccess_t	httpd_apcupsd_cgi_htaccess_t	httpd_lock_t	http_port_t	
httpd_sys_script_rw_t	httpd_apcupsd_cgi_script_rw_t	httpd_tmpfs_t	httpd_awstats_script_ra_t	
httpd_helper_exec_t	http_cache_client_packet_t	httpd_log_t	httpd_awstats_script_ro_t	
httpd_awstats_htaccess_t	httpd_awstats_script_exec_t	httpd_user_content_t	http_cache_port_t	
httpd_awstats_script_rw_t	httpd_apcupsd_cgi_script_exec_t	httpd_staff_htaccess_t	httpd_sysadm_script_rw_t	
httpd_sys_script_ra_t	httpd_prewikka_script_exec_t	httpd_suexec_exec_t	httpd_sysadm_script_ro_t	
httpd_user_script_ro_t	httpd_unconfined_script_ra_t	httpd_php_tmp_t	httpd_php_exec_t	
httpd_prewikka_content_t	httpd_prewikka_htaccess_t	httpd_staff_content_t	httpd_staff_script_ro_t	
httpd_rotatelogs_exec_t	httpd_prewikka_script_ra_t	httpd_squirrelmail_t	httpd_unconfined_script_rw_t	
http_server_packet_t	httpd_prewikka_script_rw_t	httpd_sys_htaccess_t	httpd_modules_t	
httpd_staff_script_rw_t	httpd_sysadm_script_exec_t	httpd_tmp_t	httpd_sys_script_ro_t	
httpd_sysadm_htaccess_t	httpd_staff_script_exec_t	httpd_sys_content_t	httpd_apcupsd_cgi_script_ro_t	
httpd_sys_script_exec_t	httpd_unconfined_script_exec_t	httpd_config_t	httpd_suexec_tmp_t	
httpd_sysadm_content_t	httpd_unconfined_content_t	http_client_packet_t	httpd_unconfined_htaccess_t	
httpd_sysadm_script_ra_t	httpd_apcupsd_cgi_content_t	httpd_var_lib_t	httpd_user_script_exec_t	
httpd_awstats_content_t	http_cache_server_packet_t	httpd_var_run_t		
Identified Flow-based Apache Subjects and Objects				
Apache Subjects				
applications	staff application	sysadm application	services	user application
Apache Objects				
*_node_t (10 types)	*_port_t (116 types)	*_fs_t (38types)	*_home_dir_t (4 types)	others
Identified Apache TCB(d)				
httpd_suexec_t	httpd_awstats_script_t	httpd_t	httpd_helper_t	httpd_php_t
httpd_sysadm_script_t	httpd_prewikka_script_t	httpd_rotatelogs_t	httpd_apcupsd_cgi_script_t	

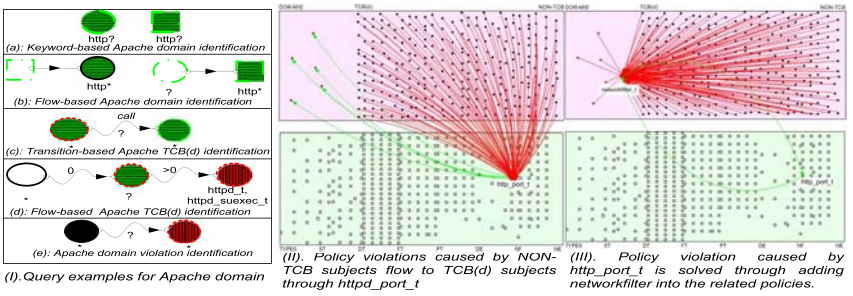


Fig. 5. Policy violations and solving

Table 2. Policy violations of Apache domain

Policy Violations			
NON-TCB	**Type:Class**	**TCB(d) Subject**	**Solve**
270	*_node_t: node	TCB(d) subjects	Filter
270	*_port_t: tcp_socket	TCB(d) subjects	Filter
270	netif_t: netif	TCB(d) subjects	Filter
6 subjects	dns_client_packet_t :packet	TCB(d) subjects	Filter
6 subjects	dns_port_t:packet	TCB(d) subjects	Filter
25	sysadm_devpts_t:chr_file	httpd_t	Modify
104	initrc_devpts_t: chr_file	httpd_t,httpd_rotatelogs_t	Modify
16	console_device_t: chr_file	httpd_t,httpd_suexec_t	Modify
270	devlog_t :sock_file	httpd_t,httpd_suexec_t	Modify
270	device_t:chr_file	TCB(d) subjects	Modify
270	devtty_t:chr_file	TCB(d) subjects	Modify
3	sysadm_tty_device_t:chr_file	httpd_t	Modify
5	urandom_device_t:chr_file	httpd_t	Modify
270	zero_device_t:chr_file	TCB(d) subjects	Modify
134	initrc_t:fifo_file	TCB(d) subjects	Modify
5	var_run_t:dir	httpd_t	Modify
72	var_log_t: dir	httpd_t	Modify
72	tmpfs_t:dir	httpd_t	Modify
httpd_staff_script_t	httpd_staff_script_*_t:file	httpd_t	Modify
httpd_user_script_t	httpd_user_script_*_t:file	httpd_t	Modify
httpd_sys_script_t	httpd_sys_script_*_t:file	httpd_t	Modify
httpd_unconfined_script_t	httpd_unconfined_script_*_t:file	httpd_t	Modify
webalizer_t	httpd_sys_content_t:file	httpd_t	Modify
httpd_apcupsd_cgi_script_t	httpd_apcupsd_cgi_script_*_t:file	httpd_t	Ignore
httpd_awstats_script_t	httpd_awstats _script_*_t:file	httpd_t	Ignore
httpd_prewikka_script_t	httpd_prewikka_script_*_t:file	httpd_t	Ignore
Further Policy Violations Example			
NON-TCB	**Type:Class**	**Adjusting Subject**	**Solve**
270	devtty_t:chr_file	httpd_prewikka_script_t	Modify
270	devtty_t:chr_file	httpd_awstats_script_t	Modify
270	devtty_t:chr_file	httpd_apcupsd_cgi_script_t	Modify

`httpd_staff_script_t` cannot be included into TCB(d) since it would lead unlimited file access for the staff services such as `staff_t`, `staff_mozilla_t`, and `staff_mplayer_t`.

Remove Policy Rules. Another way for resolving policy violations is to remove the related policy statements. For example, `webalizer_t` is to label a tool for analyzing the log files of web server and is not necessary to modify web server information. To

resolve the policy violations caused due to the write access to `httpd_sys_content_t` by `webalizer_t`, we remove the policy rule stating write_like operation between `webalizer_t` and `httpd_sys_content_t`.

Modify Policy Rules. Many policy violations are caused because related subjects or objects are given too much privileges. Hence, rather than just removing related policy statements, we also need to replace these subjects or objects with more restricted rights. For example, for policy violations caused by read and write accesses to `initrc_devpts_t`, our solution is to redefine `initrc_devpts_t` by introducing `initrc_devpts_t`, `system_initrc_devpts_t`, and `*_daemon_initrc_devpts_t` (* representing the corresponding service name). Corresponding policy rules are also modified as follows:

```
allow httpd_t initrc_devpts_t:chr_file {ioctl read getattr lock
write append}; is changed to
allow httpd_t httpd_daemon_initrc_devpts_t:chr_file {ioctl read
getattr lock write append};
```

Add Filter. Based on the domain-based integrity model, a filter can be introduced into policies to remove policy violations. For example, to remove the violations caused by `http_port_t`, we introduce a network filter subject as follows:

```
allow user_xserver_t networkfilter_t:process transition;
allow networkfilter_t http_port_t:tcp_socket {recv_msg send_msg};
```

After the modification is applied, the original policy violations are eliminated. In general, to validate the result of a policy modification, we recheck the relationships between the policy violation related domains and types. Comparing Figure 5 (c) with Figure 5 (b), we can observe that all read operations between TCB(d) and type `http_port_t` are removed. Also, the write operations between NON-TCB and `http_port_t` are also removed. Instead, a new domain `networkfilter_t` is added, which has write and read operations on `http_port_t`. Also, all TCB(d) and NON-TCB subjects can transit to this new domain type.

5 Conclusion

In this paper, we have proposed a graph-based policy analysis framework to effectively verify complex security policies for integrity protection, based on a proposed domain-based integrity model. We develop an intuitive visualization tool to demonstrate the feasibility of our framework. Additionally, we discuss how we can use our framework to analyze SELinux policies and the results demonstrate the effectiveness and efficiency of our methodology. We believe that this is the first effort to formulate a general policy analysis framework with graph-based approach. We are going to develop a violation graph based ranking schema, which can be used to help resolving the policy violations. Also, the userability of the graph-based policy analysis tool will be studied. In addition, we plan to enhance the flexibility of our approach and investigate how our policy analysis framework and tool can be utilized with other integrity models.

References

1. Tresys Technology Apol., http://www.tresys.com/selinux/
2. Trusted Computer System Evaluation Criteria. United States Government Department of Defense (DOD), Profile Books (1985)
3. Ahn, G., Xu, W., Zhang, X.: Systematic policy analysis for high-assurance services in selinux. In: Proc. of IEEE Workshop on Policies for Distributed Systems and Networks (2008)
4. Anderson, A.P.: Computer security technology planning study. Technical Report ESD-TR-73-51, II (1972)
5. Aris, A.: Network visualization by semantic substrates. IEEE Transactions on Visualization and Computer Graphics 12(5), 733–740 (2006); Senior Member-Ben Shneiderman
6. H. C. I. L. at University of Maryland. Piccolo, http://www.cs.umd.edu/hcil/jazz/download/index.shtml
7. Biba, K.J.: Integrity consideration for secure compuer system. Technical report, Mitre Corp. Report TR-3153, Bedford, Mass (1977)
8. Green, M.: Toward a perceptual science of multidimensional data visualization: Bertin and beyond (1998), http://www.ergogero.com/dataviz/dviz2.html
9. Guttman, J., Herzog, A., Ramsdell, J.: Information flow in operating systems: Eager formal methods. In: Workshop on Issues in the Theory of Security (WITS) (2003)
10. Herman, I., Melancon, G., Marshall, M.: Graph visualization and navigation in information visualization: A survey. IEEE Transactions on Visualization and Computer Graphics 6(1), 24–43 (2000)
11. Jaeger, T., Sailer, R., Zhang, X.: Analyzing integrity protection in the selinux example policy. In: Proc. of USENIX Security Symposium (2003)
12. Loscocco, P., Smalley, S.: Integrating flexible support for security policies into the linux operating system. In: USENIX Annual Technical Conference, FREENIX Track (2001)
13. Provos, N., Friedl, M., Honeyman, P.: Preventing privilege escalation. In: 12th USENIX Security Symposium, August 2003, p. 11 (2003)
14. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and implementation of a TCG-based integrity measurement architecture. In: USENIX Security Symposium (2004)
15. Sandhu, R.S.: Lattice-based access control models. IEEE Computer 26(11), 9–19 (1993)
16. Sarna-Starosta, B., Stoller, S.D.: Policy analysis for security-enhanced linux. In: Proceedings of the 2004 Workshop on Issues in the Theory of Security (2004)
17. Shankar, U., Jaeger, T., Sailer, R.: Toward automated information-flow integrity verification for security-critical applications. In: NDSS. The Internet Society (2006)
18. Smalley, S.: Configuring the selinux policy (2003), http://www.nsa.gov/SELinux/docs.html
19. Fraser, T.: Lomac: Low water-mark integrity protection for cots environment. In: Proceedings of the IEEE Symposium on Security and Privacy (May 2000)
20. WIKIPEDIA. Trusted computing base, http://en.wikipedia.org/wiki/Tusted_Computing_Base
21. Xu, W., Shehab, M., Ahn, G.: Visualization based policy analysis: case study in selinux. In: Proc. of ACM Symposium of Access Control Models and Technologies (2008)
22. Wang, H., Osborn, S.: Discretionary access control with the administrative role graph model. In: Proc. of ACM Symposium of Access Control Models and Technologies (2007)
23. Osborn, S.: Information flow analysis of an RBAC system. In: Proc. of ACM Symposium of Access Control Models and Technologies (2002)

Practical Private DNA String Searching and Matching through Efficient Oblivious Automata Evaluation

Keith B. Frikken

Department of Computer Science and Systems Analysis
Miami University, Oxford, OH 45056
frikkekb@muohio.edu

Abstract. In [18] it was shown that the ability to perform oblivious automata evaluation was useful for performing DNA searching and matching. By oblivious automata evaluation we mean that one participant has a finite state machine and the other participant has a sequence, and at the end of the protocol the sequence owner learns whether the machine accepts the sequence. A protocol was given in [18], but it required $O(n)$ rounds (where n is the number of characters in the sequence) and $O(mn)$ modular exponentiations (where m is the number of states in the automata). Both of these factors limit the applicability of this approach. In this paper we propose a new protocol that requires only $O(1)$ rounds and reduces the number of modular exponentiations to $O(n)$ without revealing any additional information. We have implemented both schemes and have shown experimentally that our scheme is two to three orders of magnitude faster than the previous scheme.

Keywords: Privacy-Preserving Protocols, DNA matching, and Cryptography.

1 Introduction

In this paper we consider the problem of evaluating an automata in an oblivious manner. That is, one participant, Alice, has an automata that she regards as private, and the other participant, Bob, has a sequence which he regards as private. We desire a protocol where Bob learns whether Alice's automata accepts his sequence, but Bob learns no other information about the automata and Alice learns no information about Bob's sequence. This problem was considered in [18] with the motivation of applying this technology to oblivious DNA searching and matching. For example, it is possible to build a finite state machine that accepts only DNA sequences that contain a specific sub-sequence that is close to a marker sequence. Assuming Alice has found such a marker (perhaps Alice is a large pharmaceutical company) and would like to sell a service to Bob where he learns whether he has a predisposition to the disease, then this technology would make it possible for Bob to learn whether he has a predisposition to the

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 81–94, 2009.

specific disease without revealing his DNA sequence to Alice or revealing Alice's proprietary information to Bob.

In the near future, it is envisioned that it will be possible to sequence one's own DNA for a modest price. For example, the US National Institute of Health has set a goal that by 2014 it should be possible to sequence a human genome for under 1000 dollars [1]. When such technology exists, genomic information in electronic form will become ubiquitous. This raises serious privacy concerns, as it is easy to envision possible abuses of such information. And while recent legislation in the US will make it illegal to discriminate based on DNA [2], it is desirable to have technology that would allow this data to be useful without having to share it with other entities.

In [18], a protocol was proposed for evaluating an automata in an oblivious manner. However, while their scheme was an excellent first step towards practical DNA searching and matching, their scheme has two performance drawbacks: i) it requires rounds linear in the size of Bob's sequence, and ii) it requires modular exponentiations proportional to the number of states in the automata times the number of characters in Bob's sequence. Since modular exponentiations require significantly more computation than many other operations, they are an accurate estimator of performance for protocols. In this paper we introduce a new scheme that reduces the number of rounds to a small constant and reduces the number of modular exponentiations to a value that is linear in the length of Bob's sequence. Furthermore, we have implemented both our scheme and the scheme in [18] and we show experimentally that our scheme is 2 to 3 orders of magnitude faster than the previous scheme.

1.1 Problem Definition/Notation

The server has a automata denoted by (Σ, S, q, M, F) where Σ is the alphabet (we denote the alphabet as $\Sigma_1, \ldots, \Sigma_{|\Sigma|}$), S is a set of states denoted by $\{s_0, \ldots, s_{m-1}\}$, $q \in S$ is the initial state, M is a $\Sigma \times m$ matrix representing the transition function, i.e., $M(i,j)$ represents the state that the automata enters when processing input Σ_i in state s_j, and $F \subseteq S$ is the set of final states. The server regards this automata as private. The client has a sequence of letters $\ell_1 \cdots \ell_n \in \{\Sigma\}^n$, that it regards as private. For convenience we denote the state that the automata is in after processing the string $\ell_1 \cdots \ell_m$ by $s_{\ell_1 \cdots \ell_m}$. More formally, $s_{\ell_1} = M(\ell_1, q)$ and $s_{\ell_1 \cdots \ell_i} = M(\ell_i, s_{\ell_1 \cdots \ell_{i-1}})$.

The goal of the protocol is for the client to learn whether the server's automata accepts its sequence (i.e., if $s_{\ell_1 \cdots \ell_n} \in F$), without revealing information about the sequence to the server (including the automata's result) and the client learns nothing other than what can be deduced from this result.

In this paper we assume that the automata owner is honest-but-curious in that it will follow the protocol exactly but will try to infer additional information. We consider this model because: i) if a protocol cannot be made practical in this restricted model then there is little hope that it could be made secure in a stronger adversary model and ii) by auditing the server and levying fines for misbehavior would give disincentives to keep the server from misbehaving.

1.2 Our Contributions

The contributions of this work are as follows:

1. We introduce a new protocol for oblivious automata evaluation that requires only a constant number of rounds (a reduction from $O(n)$ in the previous scheme), and reduces the number of modular exponentiations from $O(mn + n|\Sigma|)$ to $O(n)$.
2. We have implemented our scheme along with the scheme in [18] and we show experimentally that our proposed scheme is 2-3 orders of magnitude faster than the previous approach.
3. We extend our scheme to support transducers (i.e., finite state machines that produce more than Boolean output).

1.3 Organization of Manuscript

The remainder of this document is organized as follows. In section 2 we describe related work. In section 3 we describe tools needed for our protocol. In section 4 we describe the scheme in [18] in more detail so that it is possible to compare the schemes. In section 5 we introduce our new scheme. In section 7 we introduce experimental results, and we summarize our results in section 8.

2 Prior Work

2.1 Generic SMC

Secure Multiparty Computation (SMC) is the problem of creating a privacy-preserving protocol for any function f[1]; that is, creating a protocol that computes f over distributed inputs while revealing only the result and inferences that can be made from this result. General results state that any function can be computed in such a secure manner. The first constructions for secure two-party SMC were given in [19,20]; these assumed that the adversary of the protocol was honest-but-curious (HBC) in that the adversary will follow the protocol exactly but will attempt to make additional inferences. Later a construction was given for multiple parties [10] in the malicious adversary model (where the adversary deviates arbitrarily from the protocol) assuming that a majority of the participants are honest. There have also been many other papers attempting to improve the efficiency of these protocols to make the general results practical. Another approach that has been deployed is to introduce a domain-specific protocol that is more efficient than the general results. We describe specific secure protocols for genomic problems in the next section.

One crucial difference between the problem considered in this paper and the above-mentioned general results is that in this paper, the server's input is a function (i.e., an automata) and the function being evaluated is private. Some

[1] We are assuming that f can be computed in polynomial time when given all of the inputs.

prior approaches include: i) *Selective private function evaluation (SPFE)* and ii) Cryptocomputuing. The goal of SPFE [8] is for one party to compute a private function over a subset of another party's database without revealing the function. However, this model is different than the one used in our approach. Cryptocomputing [17] allows two parties to compute a private function on another party's input. This work was extended in [4,5] to support more general functions. In this paper we introduce a more efficient approach for the specific problem of oblivious automata evaluation.

2.2 Privacy-Preserving Protocols for Genomic Computation

The work most closely related to this paper is [18], which introduced a protocol for oblivious automata evaluation, with the goal of being able to perform robust DNA searching. We describe the protocol proposed in this paper in more detail in section 4. While there are many applications for doing oblivious automata evaluation, a primary motivation is to be able to perform DNA searching and matching. There has been other work that has considered the problem of privacy-preserving DNA matching. For example, the work in [3] introduced a protocol that computes the edit distance between two sequences and a new protocol was introduced in [11] that improved the efficiency of this scheme. Thus the work in this paper is semi-orthogonal to the work in [3,11] in that: i) for computing the edit distance between two sequences the approach described in this manuscript is less general than the other papers, but this allows the scheme to be more efficient, and ii) however, the scheme proposed in this paper allows for many other computations not supported in [3,11] in that it can evaluate an arbitrary automata.

3 Building Blocks

3.1 Oblivious Transfer

In this paper (and in [18]) we require the usage of a chosen 1-out-k Oblivious Transfer (OT). In this protocol the sender has k values $v_1, \ldots, v_k$ and the chooser has a specific index $i \in [1, k]$. At the end of the protocol the chooser obtains v_i but does not learn any information about the other values and the sender does not learn which value was chosen. We specifically, use the protocol for OT described in [15], which requires $O(1)$ rounds of communication, $O(1)$ modular exponentiations by both participants[2], $O(k)$ modular multiplications by the server and $O(k)$ communication.

3.2 Yao's Scrambled Circuit Evaluation

In Yao's Scrambled Circuit Evaluation [20], one participant will generate an oblivious circuit that computes the desired outcome and that the other

[2] Technically, the sender must perform $O(k)$ exponentiations in an initialization phase but these can be amortized over multiple runs of the protocol.

participant will evaluate this circuit. One crucial idea behind this construction is that the generator will choose two random encodings[3] for each wire of the circuit–one corresponding to 0 (false) and the other to 1(true). The evaluator learns the encoding corresponding to the actual value of the wires, but does not know what this encoding corresponds to. While the generator would know the meaning of the encoding, the generator does not know the encoding known to the evaluator. Thus neither participant knows the true value of the wire, but together they do. For a more detailed description of this protocol and a proof of security see [13].

3.3 Additively-Homomorphic Encryption

While the scheme we propose in this paper does not utilize a homomorphic encryption scheme, the scheme in [18] did, and thus to make a detailed comparison between the schemes we provide a brief overview of such encryption schemes. In an additive homomorphic encryption scheme, the product of two ciphertexts is a ciphertext of the sum of the two plaintexts. This allows basic operations to be performed on encrypted values. One specific homomorphic encryption scheme is described in [16]. More specifically, when we refer to homomorphic encryption, we are using an encryption scheme with the following properties: public-key, semantically-secure, additive homomorphism–Given $E(x)$ and $E(y)$, we require that $D(E(x) * E(y)) = x + y$ and $D(E(x)^c) = xc$, and re-encryption–Given $E(x)$ and the public parameters of the homomorphic encryption scheme it is possible to re-encrypt the value to obtain another ciphertext for x (usually this is done by multiplying by $E(0)$). Note that encryption, decryption, and re-encryption all require $O(1)$ modular exponentiations.

4 Previous Scheme and Analysis

In this section we give a high level overview of the scheme introduced in [18], and we also provide a detailed analysis of the cost of this approach. A principle idea of this scheme, is that the sequence owner (hereafter referred to as the client) will learn the current state of the automata obfuscated by an additive permutation factor chosen by the automata owner (hereafter referred to as the server). That is, the current state will be additively split between the two parties so that neither will know the actual value. Initially, the protocol is bootstrapped by having the client and server engage in a 1-out-of-$|\Sigma|$ OT protocol to learn the state of the automata after the first character of the sequence.

Protocol. To set up this protocol, it is assumed that server has received the client's public key for a semantically-secure homomorphic encryption system, and that client has received the setup information to be a chooser in an OT protocol (as in [15]).

[3] At a high level these can be thought of as cryptographic keys.

1. The server chooses n integer values, $h_1, \ldots, h_n$ where each value is chosen uniformly from $[0, m-1]$. The server constructs $|\Sigma|$ values, $v_1, \ldots, v_{|\Sigma|}$ where $v_i = M(i,q) + h_1 \bmod m$ (Recall that q is the initial state of the automata). The server and the client engage in a 1-out-of-$|\Sigma|$ OT protocol where the server inputs $v_1, \ldots, v_{|\Sigma|}$ and the client chooses ℓ_1. The client stores this value as s'_{ℓ_1} (The prime is used to denote that this is the current state additively permuted).
2. In sequence, for each value i from 2 to n they do:
   (a) The client creates a list of encrypted values $E(a'_0), \ldots, E(a'_{n-1})$ where $a'_j = 1$ if $j = s'_{\ell_1 \cdots \ell_{i-1}}$ and is 0 otherwise. The client sends these values to the server.
   (b) The server shifts these values by h_{i-1} positions to obtain $E(a_0), \ldots, E(a_{n-1})$ where $E(a_j) = E(a'_{j-h_{i-1} \bmod m})$. Note that a_j is 1 if and only if $j = s_{\ell_1 \cdots \ell_{i-1}}$. The server creates an $n \times 1$ matrix A with these values. The server then creates a $|\Sigma| \times n$ matrix M' where $M'(i,j) = M(i,j) + h_i \bmod m$. Using standard techniques for computing with homomorphically-encrypted values the server calculates $M'A$ to obtain the values $r_1, \ldots, r_{|\Sigma|}$.
   (c) The server and the client engage in a 1-out-of-$|\Sigma|$ OT protocol where the server inputs $r_1, \ldots, r_{|\Sigma|}$ and the client chooses ℓ_i. The client stores this value as $s'_{\ell_1 \cdots \ell_i}$.
3. The server creates m messages $z_1, \ldots, z_m$ where z_i is 1 if $s_{i-h_n \bmod m}$ is a final state and is 0 otherwise. The server and the client engage in a 1-out-of-m OT protocol where the server inputs $z_1, \ldots, z_m$ and the client chooses $s_{\ell_1 \cdots \ell_m}$'. If the client receives 1 then its sequence was accepted by the automata and if the client learns 0 then its sequence was not accepted by the automata.

Analysis. The above protocol requires $O(n)$ rounds as step 2 must be run sequentially. The server's computation for each letter in the sequence is $O(|\Sigma|m)$ as this is the number of modular multiplications to perform the matrix multiply and requires $O(|\Sigma|)$ modular exponentiations, thus in total the server performs $O(|\Sigma|mn)$ operations and $O(|\Sigma|n)$ modular exponentiations. The client must perform $O(m)$ operations and modular exponentiations per character in the sequence, and thus it has to perform $O(nm)$ such operations in total. Finally, the protocol requires $O(\rho_1 mn + \rho_2 n|\Sigma|)$ bits of communication, where ρ_1 is the size of an semantically-secure homomorphic encryption (e.g., 2048 bits for security comparable to 1024-bit RSA keys for the Paillier scheme [16]) and ρ_2 is the size of a hash function (e.g., 160 bits for SHA-1). Table 1 summarizes the performance of this scheme. Two performance bottlenecks with this scheme are: i) the number of rounds grows linearly with the length of the sequence being checked, and ii) the number of modular exponentiations is prohibitively large.

5 Proposed Scheme, Analysis, and Comparison

In this section we introduce our new scheme for oblivious automata evaluation; this scheme borrows ideas from Yao's Scrambled Circuit Evaluation [20]. That is, a single step of automata processing is similar to processing a single gate in

a circuit. Like the previous scheme the client will learn the current state of the automata obfuscated by an additive hiding factor. Also, for each character of the sequence the server will create an encoding[4] for each possible state and the client will learn the specific encoding corresponding to the current state. Also, for each sequence character, the server will choose an encoding for each letter of the alphabet and the client learns the encoding corresponding to his sequence's current letter. Using this permuted state and the state and alphabet encodings, we describe an oblivious transition function that allows the client to compute the next state encoding and permuted state from the previous values. More details are described below:

1. The server chooses $(n-1)|\Sigma|$ random encodings $k_{i,j}$ for $i \in [2,n]$ and $j \in [1,|\Sigma|]$. In the following, these will be referred to in the following as the alphabet encodings. In this protocol the client will learn, k_{i,ℓ_i} for each $i \in [2,n]$ and will not learn any of the other alphabet encodings. These encodings will be revealed via $n-1$ oblivious transfer protocols, which can all be done in parallel.
2. The server chooses mn random encodings $e_{i,j}$ for $i \in [1,n]$ and $j \in [1,m]$. In the following, these will be referred as the state encodings. The client will learn $e_{1,s_{\ell_1}}, e_{2,s_{\ell_1\ell_2}}, \ldots, e_{n,s_{\ell_1\cdots\ell_n}}$ and no other state encodings. That is, the client will learn one state encoding for each letter of his sequence, and this state encoding will be the one corresponding to the actual state of the automata after processing that portion of the client's sequence.
3. The server will choose n state permutation values $h_1, \ldots, h_n$ and the client will learn $s_{\ell_1} + h_1 \bmod m$, $s_{\ell_1\ell_2} + h_2 \bmod m, \cdots, s_{\ell_1\cdots\ell_n} + h_n \bmod m$. That is, the client will learn the obfuscated state of the automata after processing each letter.

We will now describe an oblivious transition function that maps $(k_{g,j}, e_{g-1,s}, s+h_{g-1} \bmod M)$ to $(e_{g,M(j,s)}, M(j,s)+h_g \bmod M)$. That is if after $g-1$ characters the client is in state s, and its gth letter was Σ_j, then this oblivious function allows the client to learn the encoding for $M(j,s)$ (i.e., the next state of the automata) along with the value of this state permuted by the hiding factor. We will denote this gth transition function as a $|\Sigma| \times m$ matrix as G_g, where

$$G_g[j,i] = (e_{g,M(j,s)} || M(j,s) + h_g \bmod m) \oplus f(e_{g-1,s}, k_{g,j})$$

where $s = i - h_g \bmod m$ and f is a pseudorandom function[5]. Note that in order to obtain the value $(e_{g,M(j,s)} || M(j,s) + h_g \bmod m)$ the client must have both $e_{g-1,s}$ and $k_{g,j}$, and since the client has at most one state encoding for $g-1$ and one alphabet encoding for g it will be able to only obtain one state encoding for g.

[4] Hereafter when referring to the term encoding, we are referring to a value chosen from $\{0,1\}^\rho$ for some security parameter ρ.

[5] Note this idea is very similar to the ideas used in [20,14] in that these scheme built an oblivious gate evaluation using similar encodings/ permuted values. Furthermore, such schemes were proven secure in [13].

Now suppose the client has G_g and that $s = s_{\ell_1 \ldots \ell_{g-1}}$. Further suppose that the client has $i = s + h_{g-1} \bmod m$, $e_{g-1,s}$, k_{g,ℓ_g}, and ℓ_g the client can compute $G_g[\ell_g, i] \oplus f(e_{g-1,s}, k_{g,\ell_g})$ to obtain $e_{g,M(\ell_g,s)}$ and $M(\ell_g, s) + h_g \bmod m$.

To help clarify the above process, we now give an example which demonstrates the above protocol. Suppose the server has an automata with two states that processes an alphabet of size two where the automata transition function is as follows: $M[0,0] = 0, M[0,1] = 1, M[1,0] = 0, M[1,1] = 1$. Further suppose that h_{g-1} is 1 and h_g is 0, then the values of G_g are as follows:

- $G_g[0,0] = e_{g,0}||0 \oplus f(e_{g-1,1}, k_{g,0})$
- $G_g[0,1] = e_{g,1}||1 \oplus f(e_{g-1,0}, k_{g,0})$
- $G_g[1,0] = e_{g,0}||0 \oplus f(e_{g-1,1}, k_{g,1})$
- $G_g[1,1] = e_{g,1}||1 \oplus f(e_{g-1,0}, k_{g,1})$

Further suppose that the client is in state 0 after $g-1$ characters and that his next character is 1. Now the client has the following values: $e_{g-1,0}$ (the encoding corresponding to its state), $i = 1$ which is its current state permutes with h_{g-1}, and $k_{g,1}$ (its alphabet encoding for ℓ_g). The client computes $G_g[\ell_g, i] \oplus f(e_{g-1,0}, k_{g,1}) = G_g[1,1] \oplus f(e_{g-1,0}, k_{g,1}) = e_{g,1}||1$ and thus it learns its new permuted state. Note that the permuted state is 1, and since the new hiding factor is 0, this corresponds to the automata being in state 1.

In what follows we describe the protocol in detail:

1. The client and the server engage in $(n-1)$ 1-out-of-$|\Sigma|$ OTs (in parallel) where in the ith protocol the server acts as the sender and uses $k_{i,\Sigma_1}, \ldots, k_{i,\Sigma_{|\Sigma|}}$ and the client inputs ℓ_i for $i \in [2,n]$. After this step the client will have k_{i,ℓ_i} for all $i \in [2,n]$.
2. The client and server engage in 1-out-of-$|\Sigma|$ OT where the server inputs $a_1, \ldots, a_{|\Sigma|}$ where $a_i = e_{1,M(\Sigma_i,q)}||M(\Sigma_i, q)+h_1 \bmod m$ and the client inputs ℓ_1. After this step the client will have the permuted state and the encoding for the automata after processing the client's first character.
3. The server calculates $G_2, \ldots, G_n$ and sends them to the client. The client then evaluates the functions in sequence as described above and after evaluating G_n we will denote the permuted state by $f = s_{\ell_1 \ldots \ell_n} + h_n \bmod m$.
4. The server creates m messages $z_1, \ldots, z_m$ where z_i is $Enc(1, e_{n,i-h_n \bmod m})$ if $s_{i-h_n} \bmod m \in F$ (i.e., it is a final state) and is $Enc(0, e_{n,i-h_n \bmod m})$ otherwise [6]. Note that by encrypting the result with the final encoding this prevents the client from choosing an arbitrary state to learn information about the automata. The server and the client engage in a 1-out-of-m OT protocol where the server inputs $z_1, \ldots, z_m$ and the client chooses f. The client decrypts the value with the encoding of the final state and if the client receives 1 then its input was accepted by the automata and if the client learns 0 then it was not accepted by the automata.

Analysis. Note that the above protocol can be done in $O(1)$ rounds. Also, the server needs to perform $O(mn|\Sigma|)$ work to calculate all of the oblivious transition

[6] We are denoting the encryption of a message m with a key k was $Enc(M, k)$.

functions, but only needs to perform $O(n)$ modular exponentiations (to perform OTs). Similarly the same amount of computation and modular exponentiations need to be performed by the client. Finally, the size of the oblivious transition function information is $O(\rho_2 mn|\Sigma|)$ (where ρ_2 is the size of a secure pseudorandom function) and to perform the n OTs the communication requirements are $O(\rho_1 n\Sigma)$ (where ρ_1 is the size of a modulus where the discrete logarithm problem is hard).

The careful reader may be wondering why the above scheme does not simply just use Yao's garbled circuit evaluation to compute the state of the automata. That is, what is the efficiency of building a circuit that evaluates the automata? The efficiency of this approach is a function depends on two factors: i) the number of inputs into the circuit, and ii) the number of gates in the circuit. Clearly, the number of inputs would be $O(n \log |\Sigma|)$ as each of the n letters has $O(\log \Sigma)$ bits. The evaluation of each state is essentially a table lookup in a table with $O(m|\Sigma|)$ entries. Thus in the straightforward circuit, each letter will require $O(m|\Sigma|)$ equality checks each involving $O(\log m + \log |\Sigma|)$ bits. And so, each state would require $O(m|\Sigma|(\log m + \log |\Sigma|))$ computation, and thus the total computation would be $O(mn|\Sigma|(\log m + \log |\Sigma|))$.

Table 1 summarizes the performance of all three approaches. We now compare the scheme in this paper and the scheme in [18]: Clearly, the number of rounds required by our scheme is substantially improved in our scheme. Furthermore, we have essentially replaced the modular exponentiations of the previous protocol with the evaluation of a pseudorandom function. And since pseudorandom functions are two to three orders of magnitude faster than homomorphic encryptions, we would expect to see a large performance difference between these schemes. Finally, it might appear that the communication required by our protocol is higher than that of the previous scheme, for the specific problem of DNA matching (i.e., when $|\Sigma| = 4$), then $\rho_2|\Sigma| \leq \rho_1$, so the communication between these schemes is comparable. Also, the performance of the proposed scheme is asymptotically superior to the generic construction based on Yao's protocol.

Security Analysis. In the full version of the paper we will provide a detailed security proof, but we give only a brief overview here. The basic argument rests on the composition theorem from [7], that is when we replace the OT protocols and oblivious gate evaluation function with ideal versions (that utilize a Trusted Third Party) of the protocol it is straightforward to show that the resulting protocol is secure. Thus when we replace these functions with secure protocols, the

Table 1. Performance Comparison Summary

Metric	**Original [18]**	**Proposed**	**Yao-based**
Rounds	$O(n)$	$O(1)$	$O(1)$
Server Computation	$O(mn\|\Sigma\|)$	$O(mn\|\Sigma\|)$	$O(mn\|\Sigma\|(\log m + \log \|\Sigma\|)))$
Server Mod Exps	$O(n\|\Sigma\|)$	$O(n)$	$O(n \log \|\Sigma\|)$
Client Computation	$O(mn)$	$O(mn)$	$O(mn\|\Sigma\|(\log m + \log \|\Sigma\|)))$
Client Mod Exps	$O(mn)$	$O(n)$	$O(n \log \|\Sigma\|)$
Communication	$O(\rho_1 mn + \rho_2 n\|\Sigma\|)$	$O(\rho_1 n\|\Sigma\| + \rho_2 mn\|\Sigma\|)$	$O(\rho_2 mn\|\Sigma\|(\log m + \log \|\Sigma\|)) + \rho_1 n \log \|\Sigma\|)$

resulting protocol is also secure. More specifically, we will first utilize ideal implementations of Oblivious Transfer (OT) and of Oblivious State Evaluation (OSE), specifically, OT will allow the client to input a value to a TTP $\ell \in [1, |\Sigma|]$ and the server to input Σ values $k_1, \ldots, k_\Sigma$ to the TTP. After receiving all of these values the OT TTP sends to the client the values k_ℓ and the server receives $\perp$. In the ideal OSE protocols, the client sends $(k_{g,j}, e_{g-1,s}, s+h_{g-1} \bmod M)$ and the server sends $(k_{g,1}, \ldots, k_{g,|\Sigma|}, e_{g-1,1}, \ldots, e_{g-1,n}, e_{g,1}, \ldots, e_{g,n}, M, h_g, h_{g+1})$ and after receiving all of the inputs the TTP sends the client $(e_{g,M(j,s)}, M(j,s)+h_g \bmod m)$ and sends the server $\perp$. Note that the ideal OT functionality can be achieved with the protocol in [15] and the ideal OSE functionality follows from [13].

6 Extensions

6.1 Efficiency Improvements

In this section we propose three techniques to further improve the efficiency of our approach.

Communication Reduction. By utilizing Private Information Retrieval (PIR), it is possible to reduce the communication to something sub-linear in mn, however this performance increase comes at a cost of increasing the rounds to $O(n)$. Recall that in PIR a sender has a database of values and a chooser learns at least one of the items of his choosing from the database without the sender learning which value was chosen. Single server solutions for PIR exist that require sub-linear communication [9,12,6]. To integrate this with our scheme, notice that when processing G_g in Step 3, the client only needs the one specific value from this function (the one corresponding to the permuted state and current letter). Thus the client and server could engage in a PIR protocol where the client learns the value that it needs to move to the next state of the automata.

Modular Exponentiation Reduction. Using the techniques described in [15] it is possible to reduce the number of modular exponentiations. The principle idea behind this is that it is possible to replace the n 1-out-of-$|\Sigma|$ OTs with n/k 1-out-of-$|\Sigma|^k$ OTs (i.e., we merge k OTs together). This reduces the number of modular exponentiations to n/k, but it increases the computation/communication due to OTs to $\frac{n|\Sigma|^k}{k}$. However for small $|\Sigma|$ and k this approach may improve the performance of the scheme, because the additional expense is outweighed by performance gain from removing the modular exponentiations.

Precomputation. Another advantage of the scheme proposed in this paper over the scheme in [18] is that in this scheme the server can precompute a large portion of its work. More specifically, in our scheme the server can precompute all of the oblivious transition functions as they do not depend on the client's input. However, in the scheme outlined in [18], the server must wait until it receives the client's homomorphic encryption key (which is different for every client) before it can do any of the operations for that client.

6.2 Extending Protocol to Support Transducers

In this section we describe modifications to our protocol that allow us to evaluate a transducer instead of an automata, note that the protocols in [18] also had such a generalization. In what follows we describe schemes for both Moore machines and Mealy machines.

Moore Machines. At a high level a Moore machine is a automata that produces an output vector with one character of output for each sequence character and furthermore the output is determined by the current state of the machine (i.e., each state produces a specific output character). More formally, a Mealy machine is a 6-tuple $(\Sigma, \Pi, S, q, M, \lambda)$ where Σ, S, q, M have the same meaning as in a automata, Π is the output alphabet, and $\lambda : S \rightarrow \Pi$ maps each state to an ouptut symbol. In such a protocol, if the machine is in states $t_1, \ldots, t_n$ when processing the sequence, the client will learn $\lambda(t_1) \cdots \lambda(t_n)$[7]

To modify our protocol to support such machines, the server will include the output character in the oblivious transition function. More specifically, it will change $G_g[j, i] =$

$$(e_{g,M(j,s)}||M(j,s) + h_g||\lambda(M(j,s))) \oplus f(e_{g-1,s}, k_{g,j})$$

where $s = i - h_g \bmod m$. Of course the pseudorandom function would have to be expanded to include enough bits to encrypt the output symbol. Another change to the protocol is that the final step which decodes the final state can be omitted.

Mealy Machines. The principle difference between a Mealy machine and a Moore machine is that a Mealy machines output depends on the state and the current character of the sequence. More formally, a Moore machine is a 6-tuple $(\Sigma, \Pi, S, s, M, \lambda)$ where Σ, S, s, M have the same meaning as in a automata, Π is the output alphabet, and $\lambda : S \times \Sigma \rightarrow \Pi$ maps each state and character combination to an output symbol. The only change that needs to be done is to change $G_g[j, i] =$

$$(e_{g,M(j,s)}||M(j,s) + h_g||\lambda(M(j,s),j)) \oplus f(e_{g-1,s}, k_{g,j})$$

(i.e., we change $\lambda(M(j,s))$ to $\lambda(M(j,s),j)$.

7 Experimental Evaluation

We implemented both the protocol in [18] and the one described in this paper. The implementation was in Java, and the experiments were run on two Dell machines with 2.0 GHz Intel Core Duo processors and 512MB of RAM machines connected in a LAN setting. For all experiments we set the size of the alphabet to be 4. In the first experiment we ran we set the size of the client's sequence to 10

[7] It is straightforward to modify this protocol so that this output sequence is additively split between the client and the server.

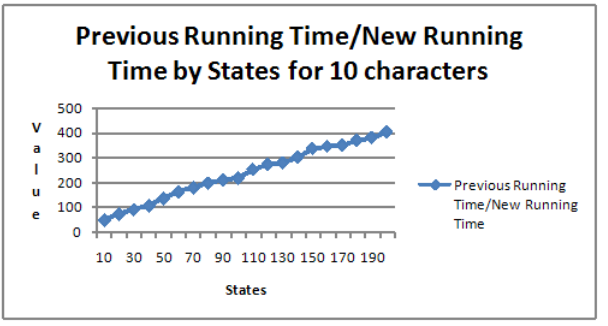


Fig. 1. Performance Difference by States

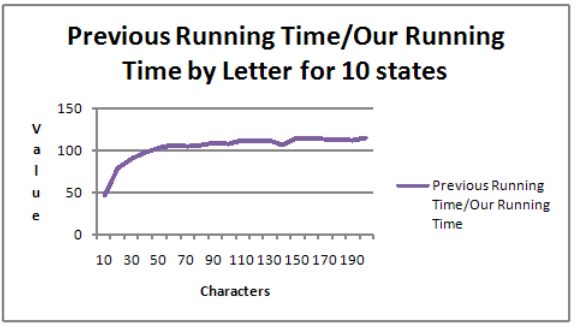


Fig. 2. Performance Difference by Letters

and we varied the states in the server's automata from 10 to 200. We ran each test 10 times. In figure 1 we show the performance difference between the previous schemes and our scheme. By this we mean this is the value of (previous scheme's running time)/(our scheme's running time). Notice that as the number of states increases the performance difference also appears to increase. Also notice that for 200 states our scheme is about 400 times faster than the previous approach.

In the second experiment we set the number of states to be 10 and varied the number of letters from 10 to 200. We also ran these tests 10 times. Like the previous test we considered the performance difference between the two schemes. Figure 2 shows the results. Notice that as the number of letters gets large our scheme is about 100 times better, but that this also appears to level off at this value. This would be expected because the number of modular exponentiations in our scheme is $O(n)$, and thus n has a significant impact on performance for our scheme.

In our final experiment (see Figure 3) we consider the running time in milliseconds for our scheme on 10 characters but we vary the number of states from 500 to 10000. Notice that to process 10 characters in a 10000 state automata our scheme seems to require about 8 seconds. When we extrapolate our data for the previous scheme we estimate that the previous scheme would take about

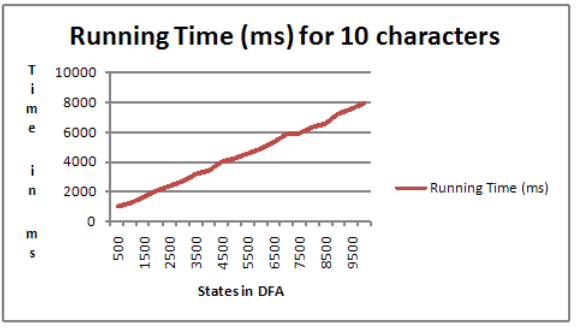


Fig. 3. Performance of Our Scheme for 10 characters by States

4 hours to handle this size of automata (however due to the length of such an experiment we did not run this test to verify if this was true). To put this in perspective, according to [18] an automata with 2000 states can be built that accepts all sequences which contain a subsequence with edit distance 2 (or less) from another sequence of length 50.

8 Summary

In this paper we introduced a scheme for oblivious automata evaluation. Previous work in this area has highlighted the importance of this problem when doing DNA searching and matching. The advantages of our approach is that we reduce the rounds to $O(1)$ and we significantly reduce the number of modular exponentiations. This turns out to be significant in practice as we show experimentally that our scheme is 2 to 3 orders of magnitude faster than the scheme in [18].

References

1. The 100 dollars Genome,Technology Review, published by MIT, April 17 (2008), `http://www.technologyreview.com/Biotech20640/page1/`
2. Genetic Information Nondiscrimination Act, `http://en.wikipedia.org/wiki/Genetic_Information_Nondiscrimination_Act`
3. Atallah, M., Kerschbaum, F., Du, W.: Secure and private sequence comparisons. In: WPES 2003: Proceedings of the 2003 ACM workshop on Privacy in the electronic society, pp. 39–44. ACM, New York (2003)
4. Beaver, D.: Minimal-latency secure function evaluation. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 335–350. Springer, Heidelberg (2000)
5. Cachin, C., Camenisch, J., Kilian, J., Müller, J.: One-round secure computation and secure autonomous mobile agents. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, p. 512. Springer, Heidelberg (2000)
6. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)

7. Canetti, R.: Security and composition of multiparty cryptographic protocols. Journal of Cryptology 13(1), 143–202 (2000)
8. Canetti, R., Ishai, Y., Kumar, R., Reiter, M., Rubinfeld, R., Wright, R.: Selective private function evaluation with applications to private statistics. In: Proceedings of the twentieth annual ACM symposium on Principles of distributed computing, pp. 293–304. ACM Press, New York (2001)
9. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM 45(6), 965–981 (1998)
10. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the nineteenth annual ACM conference on Theory of computing, pp. 218–229. ACM Press, New York (1987)
11. Jha, S., Kruger, L., Shmatikov, V.: Towards practical privacy for genomic computation. In: IEEE Symposium on Security and Privacy, SP 2008, May 2008, pp. 216–230 (2008)
12. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: FOCS 1997: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS 1997), pp. 364–373. IEEE Computer Society, Los Alamitos (1997)
13. Lindell, Y., Pinkas, B.: A proof of yao's protocol for secure two-party computation. Cryptology ePrint Archive, Report 2004/175 (2004), `http://eprint.iacr.org/`
14. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay – a secure two-party computation system. In: Proceedings of Usenix Security (2004)
15. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: SODA 2001: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, pp. 448–457. Society for Industrial and Applied Mathematics (2001)
16. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
17. Sander, T., Young, A., Yung, M.: Non-interactive cryptocomputing for NC1. In: 40th Annual Symposium on Foundations of Computer Science, pp. 554–566 (1999)
18. Troncoso-Pastoriza, J., Katzenbeisser, S., Celik, M.: Privacy preserving error resilient dna searching through oblivious automata. In: CCS 2007: Proceedings of the 14th ACM conference on Computer and communications security, pp. 519–528. ACM, New York (2007)
19. Yao, A.: Protocols for secure computations. In: Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, pp. 160–164. IEEE Computer Society Press, Los Alamitos (1982)
20. Yao, A.: How to generate and exchange secrets. In: Proceedings of the 27th IEEE Symposium on Foundations of Computer Science, pp. 162–167. IEEE Computer Society Press, Los Alamitos (1986)

Privacy-Preserving Telemonitoring for eHealth

Mohamed Layouni[1], Kristof Verslype[2], Mehmet Tahir Sandıkkaya[3], Bart De Decker[2], and Hans Vangheluwe[1]

[1] School of Computer Science, McGill University, Montreal, Canada
[2] Department of Computer Science, K.U. Leuven, Leuven, Belgium
[3] Katholieke Hogeschool Sint-Lieven, Gent, Belgium

Abstract. Advances in communication technology have opened a myriad of new possibilities for the remote delivery of healthcare. This new form of service delivery, not only contributes to the democratization of healthcare, by reaching far-away populations, but also makes it possible for elderly and chronically-ill patients to have their health monitored while in the comfort of their homes. Despite all of these advantages, however, patients are still resisting the idea of medical telemonitoring. One of the main obstacles facing the adoption of medical telemonitoring, is the concern among patients that their privacy may not be properly protected. We address this concern, and propose a privacy-preserving telemonitoring protocol for healthcare. Our protocol allows patients to selectively disclose their identity information, and guarantees that no health data is sent to the monitoring centre without the patients' prior approval. The approval process can be automated, and requires only an initial configuration by the patient.

1 Introduction

The phenomenal medical advances achieved in recent years, as well as the remarkable improvements in overall quality of life, have led to a significant increase in lifespan. This increased longevity is directly reflected in the worldwide emergence of a larger elderly and chronically ill population—a population in need of a special, sometimes round-the-clock, geriatric care. The growth of this type of patient population represents a whole new challenge to existing healthcare infrastructures worldwide. In order to deal with this challenge, countries around the world have experimented with a variety of approaches [Can08, US08, Nag06, AMD08]. One of the most promising among these approaches, is the adoption of telemonitoring. Telemonitoring is the medical practice of remotely monitoring the health of patients in the comfort of their homes, or more generally, outside traditional healthcare environments (e.g., hospitals, nursing homes, doctor's office). The idea of medical telemonitoring itself is not new [Nag06]. In 1906, Willem Einthoven, father of electrocardiography, sought to transmit electrodiagrams over telephone lines [Ein08]. In the 1920s, ship radios were used to link doctors with sailors to assist during medical emergencies at sea. In the 1970s, paramedics in remote Canadian villages were able to perform

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 95–110, 2009.

life-saving interventions while linked with hospitals in distant towns via satellite. Today, telemedicine is beginning to mature considerably with new advances in communication technology. Although telemedicine is not suitable for all patients in all cases, it is still considered highly advantageous in many situations. For instance, medical telemonitoring has been successful in helping elderly people avoid nursing homes, maintain a more dignified social life, remain productive, stay home longer, and thus incur less healthcare costs. Medical telemonitoring has also helped decrease the burden on the country's healthcare infrastructure and economy as a whole. Besides the above aspects, medical studies have shown that telemonitoring makes patients more health-conscious as a result of being personally engaged in the health monitoring process. The study in [TSI+08] for example, found that patients became significantly more proactive and careful about their health, after being required to take their own blood pressure at regular time intervals.

Despite all the above advantages, patients are still showing a certain reluctance to accept the idea of medical telemonitoring. This lack of acceptance is generally attributed to two main concerns:

1. **Dependability.** How efficient is telemonitoring compared to an in-person visit to the hospital? Will someone be there to help in case of emergency?
2. **Privacy.** Is private data properly protected when sent over the wires to the hospital?

In this paper, we assume that sufficient resources are available to make the system dependable, and focus our attention on solving the privacy aspect of the problem. In particular, we present a medical telemonitoring protocol that preserves the patients' privacy.

Organization. We start in Section 2 by describing our system setting and model. In Section 3, we present the list of security and privacy requirements that our protocol achieves. We then give a high-level overview of our solution, along with a description of the building blocks, in Sections. 4 and 5. The details of the proposed protocol are highlighted in Section 6. In Section 7, we discuss the security and privacy features of the proposed protocol. In Section 8, we highlight related work before concluding in Section 9.

2 System Model and Settings

We assume a setting where two main parties are involved: a patient at home, and a health monitoring centre (*HMC*) located at the hospital. We assume that the patient has two types of devices:

– Measurement devices : they are used to measure the patient's vital signs such as his heart rate. Measurement devices can be portable wearable devices that patients carry on them all the time (e.g., a wearable heart-rate monitor in the form of a bracelet or a belt [Zep08]), or stationary static devices that the patient can use at discrete points in time (e.g., a conventional tensiometer)

– A master storage and communication device $\mathcal{M}$: this device, usually located at the patient's home, collects information from the measurement devices[1].

The collected information is stored and analyzed by the master device $\mathcal{M}$. Based on this analysis, $\mathcal{M}$ sends a summary report about the patient's condition to the health monitoring centre periodically. However, in case the collected measurements indicate a sudden health deterioration, or a condition that requires immediate attention, the master device automatically triggers an emergency signal and sends an immediate notification to the health monitoring centre. In response to this notification, and following the specific medical practices in place, the health monitoring centre may send an ambulance to the patient's home, or notify the patient's neighbours and family members, etc. Figure 1 gives a summary of the overall setting.

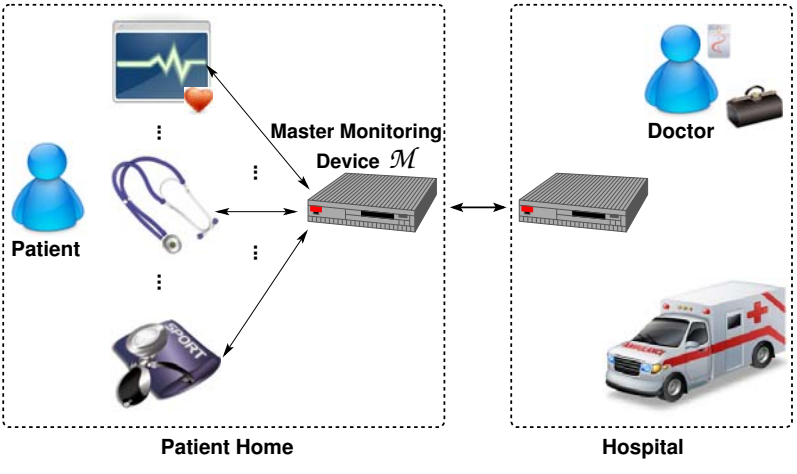


Fig. 1. Overview of the Health Telemonitoring System

From a design point of view, several issues need to be addressed in such a system. For example, the system has to be easy to use; the communications between the measurement devices and the master device on one hand, and between the master device and the health monitoring centre on the other hand, have to be seamless to the patient. The system has to be user-friendly; for example if the patient does not feel well he should be able to *easily* reach out for an emergency button to call for help. More importantly, the system has to be reliable, and the exchange of information should not compromise the patient's privacy.

In this paper, we focus on the security and privacy aspects of the system, and propose a construction that ensures a set of properties that we describe in the following section.

[1] For portable devices this transfer of information takes place once the patient is back at home.

3 Security and Privacy Requirements

- **Selective disclosure.** This property captures the ability of the patient to wilfully disclose *fine-grained* information about his identity attributes, and hide the rest. Selective disclosure implies another useful property called *minimal disclosure*, which describes a user's ability to disclose the minimum information necessary for a transaction to take place. For example, in order to receive the services of a local hospital, the minimum information required could be a proof that the patient is a resident of a certain postal code area, without the need to reveal his exact address.
- **Patient-centricity.** A system is said to be *patient-centric* if and only if it guarantees that any data disclosed to the monitoring centre, must have received the prior approval of the patient.
- **Pseudonimity.** A system is said to preserve pseudonimity if data records sent from the patient's home to the monitoring centre are linkable to each other (*e.g.,* via a patient's pseudonym) but not to the patient's real identity.
- **Conditional deanonymization.** In cases of emergency, it should be possible to recover the real identity behind the patient's pseudonym, so that urgent help can be provided.
- **Integrity.** It should not be possible to alter health information on the way between the measurement sensors and the master device, or between the master device and the monitoring centre, without being detected.
- **Confidentiality.** The content of the data sent by the master device should be readable only by the monitoring centre, as intended.

4 Solution Outline

First, we assume that the patient has one or more measurement devices and one master monitoring device $\mathcal{M}$. The measurement devices communicate only with the master device $\mathcal{M}$, which in turn communicates with the health monitoring centre. The measurement and master monitoring devices are issued by the health monitoring centre to the patient. Both types of devices are assumed to be tamper-resistant. Each device is assigned some key material before being handed over to the patient. The key assignment process includes the embedding of some key material into a protected memory location on the device.

Without loss of generality and to keep the presentation simple, we assume in the remainder, that the patient has a single measurement device m, in addition to the master monitoring device $\mathcal{M}$. Let k_m be a symmetric encryption key assigned to the measurement device m, and shared with the master device $\mathcal{M}$. Let $(PK_{\mathcal{M}}, SK_{\mathcal{M}})$ be the public/private-key pair assigned to $\mathcal{M}$.

The measurement device m collects health readings from the patient and sends them encrypted under k_m to the master device $\mathcal{M}$. To ensure the integrity of messages between m and $\mathcal{M}$, the encrypted information can be signed using a message authentication code (MAC). Alternatively, conventional public key signatures (e.g., RSA) can be used. However, the former option is the more efficient and preferred one.

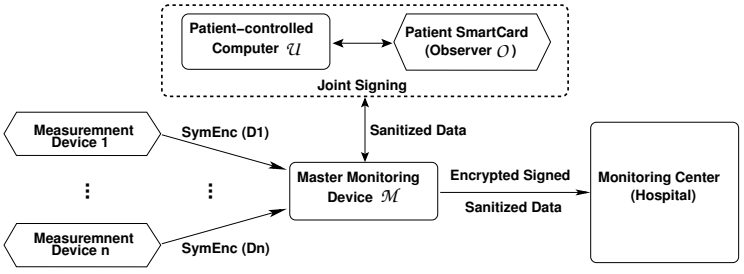


Fig. 2. Interaction between the Master Monitoring Device and the Hospital

The master device first analyzes the collected measurements, and removes any identifying information from them using data sanitization techniques (see Sec. 5.2.) The sanitized data is then signed by a computer under the patient's control, before being sent encrypted to the hospital. The use of a patient-controlled computer, that is separate from the master device $\mathcal{M}$, to sign health data, is intended to strengthen *user-centricity*. This design choice also helps guarantee that data releases have been approved by the patient. User-centricity can be further enhanced by using a two-factor message authentication mechanism, where computing signatures requires both the knowledge of a secret key (stored on the patient's computer) and the possession of a valid smartcard.

The signing procedure does not require the direct intervention of the patient, and can be configured to work automatically.

Privacy-preserving two-factor message authentication. To ensure patient-centricity as well as selective disclosure capabilities, we use the wallet-based version of Brands credential system (WBr) [Bra00, Chap 6]. The WBr system provides a way to authenticate communications between the master device and the health monitoring centre, while allowing the patient to selectively disclose his identity information [Bra00, Sec 6.3]. The WBr is a two-factor authentication system, where performing signatures on data, requires the patient to be in possession of a valid smartcard and to know the secret attributes underlying his credential. This feature makes attacks by impersonation against the patient harder, even when the patient's computer is compromised.

In our context, the master device sanitizes the patient data, and then sends it to the patient-controlled computer for signing. The latter signs the data using the patient's anonymous credential. The signed data is then sent over to the hospital, encrypted under the monitoring centre's public key. Owing to the "selective disclosure" capabilities of anonymous credentials, the patient is able to wilfully disclose, via the computed signature, any self-approved information or property about his identity, while keeping everything else private.[2] The WBr system has a built-in mechanism to block any possible covert channels between the patient's smartcard and the outside world (e.g., monitoring centre.) This

[2] *Selective disclosure* cannot be achieved using conventional X.509 certificates.

mechanism represents an additional guarantee that no data about the patient's identity will be sent without his approval. Figure 2 shows a high-level overview of the interaction between the master device and health monitoring centre.

5 Building Blocks

5.1 Brands Wallet-Based Anonymous Credentials

In [Bra00, Chap 6], Brands presents a credential system (WBr) suitable for the *wallet-with-observer* setting, where the user holds a *wallet* composed of a self-controlled computer denoted $\mathcal{U}$, and a tamper-proof device called *Observer* and denoted $\mathcal{O}$. The observer is supplied by a recognized certification authority CA. The WBr system is such that credentials are issued to the pair $(\mathcal{U}, \mathcal{O})$, and can only be shown if the two entities approve the showing, and perform it jointly. The WBr system provides a number of privacy preserving features including anonymity and selective disclosure.

Settings and Assumptions. The WBr system operates in the Discrete Logarithm setting, and its security is based on the DL assumption [Bra00, Chap 6]. Let p and q be two large primes such that $2q|(p-1)$. Let G_q be the unique subgroup of $\mathbb{Z}_p^*$ of order q, and let g_0 be one of its generators. In the setup phase, the CA randomly chooses $y_1, \cdots, y_\ell \in_R \mathbb{Z}_q$ and $x_0 \in_R \mathbb{Z}_q^*$, and computes $(g_1, \cdots, g_\ell, h_0) := (g_0^{y_1}, \cdots, g_0^{y_\ell}, g_0^{x_0}) \mod p$. Next, the CA chooses a collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. Finally, the parameters G_q, H, $(g_0, g_1, \cdots, g_\ell, h_0)$, $(g_1^{x_0}, \cdots, g_\ell^{x_0}, h_0^{x_0})$ are all made public.

Issuing Credentials to a Wallet-with-Observer. A credential consists of a public key h and a signature on it, $\sigma_{\mathrm{CA}}(h)$, issued by a certification authority CA. Let $(h, \sigma_{\mathrm{CA}}(h))$ denote a credential issued by the CA to a wallet $(\mathcal{U}, \mathcal{O})$. Let $x_1, x_2, x_3 \cdots, x_{\ell'}, x_{\ell'+1}, \cdots, x_\ell$ denote the attributes embedded in h. The attributes are such that x_1, x_2 are known only to the wallet observer $\mathcal{O}$, $x_3 \cdots, x_{\ell'}$ and x_ℓ are known only to the user-controlled computer $\mathcal{U}$, while $x_{\ell'+1}, \cdots, x_{\ell-1}$ are known both to the CA and $\mathcal{U}$. The fact that attributes x_1, x_2 are known only to the wallet observer $\mathcal{O}$, and that $x_3 \cdots, x_{\ell'}$ and x_ℓ are known only to the user-controlled computer $\mathcal{U}$, means that the issued credential can be shown to a verifier only if $\mathcal{O}$ and $\mathcal{U}$ both approve the showing and participate in it. Figure 3 depicts the issuing process.

At the start of the issuing protocol, observer $\mathcal{O}$ computes a message M_1 containing a number of commitments and a signed proof of knowledge. M_1 is sent to $\mathcal{U}$ and then forwarded to the CA.

To provide guarantees that M_1 is originating from a legitimate observer, we assume that all legitimate observers have CA-supplied certificates embedded in them. These certificates are independent from the wallet holder's identity, and their embedding takes place before the wallets are attributed to a particular user. The certificates are of the form $(e := g_{\mathcal{O}}^x, cert_{\mathrm{CA}}(e))$, where $g_{\mathcal{O}}$ is a public generator of G_q, and x a secret randomly chosen by the observer in question.

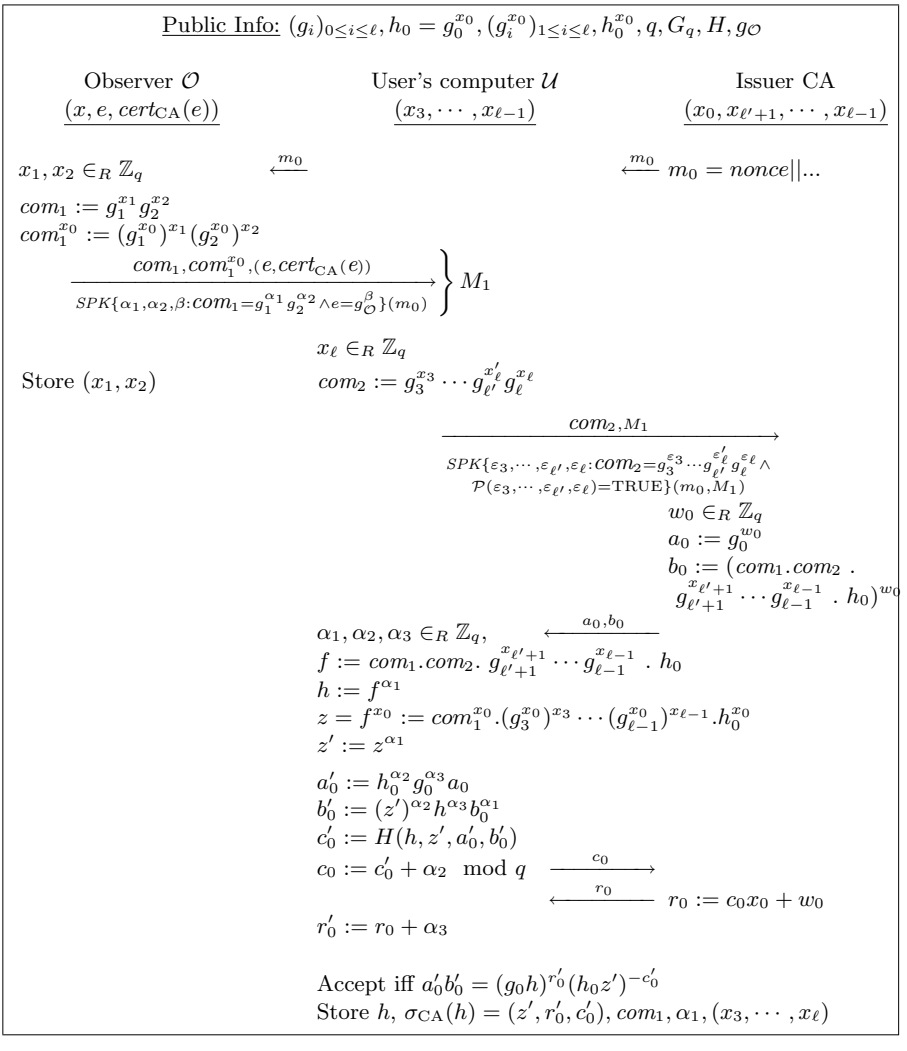


Fig. 3. Protocol for Issuing a Credential to a Wallet-with-Observer (NB. Attributes $x_{\ell'+1}, \cdots, x_{\ell-1}$ are known to both $\mathcal{U}$ and the CA. The secret key x, underlying the Observer's certificate $(e, cert_{\mathrm{CA}}(e))$, is such that $e := g_{\mathcal{O}}^{x}$).

A legitimate observer authenticates its messages, by including its certificate in M_1, and proving knowledge of the underlying secret x.

It is worth noting that the issuing protocol shown in Figure 3 is a blind signature protocol, since the CA does not learn any information about the credential $(h, \sigma_{\mathrm{CA}}(h))$ obtained by the pair $(\mathcal{U}, \mathcal{O})$. More details on the *blinding* features of the protocol can be found in [Bra00, Chap 6].

It is also worth mentioning that in the basic WBr system, the observer holds only *one* secret x_1, as opposed to two secrets (x_1, x_2) as in the protocol of Figure 3. We made the latter choice to protect the the observer's secrets (x_1, x_2) *unconditionally*, even from the user-controlled computer $\mathcal{U}$. This is achieved because given the commitment $com_1 := g_1^{x_1} g_2^{x_2}$, there are q solutions $(X, Y) \in (\mathbb{Z}_q^*)^2$ to the equation $com_1 = g_1^X g_2^Y$. Because these solutions are all equiprobable, even an all-powerful $\mathcal{U}$ cannot guess which one is encoded in com_1, with a probability of success better than $1/q$. Therefore the observer's secrets are *unconditionally* protected. This technique is similar to those used by Okamoto in [Oka92].

Issuing protocol correctness (sketch). By examining the protocol, we can derive the following equalities:

- $a'_0 = g_0^{(\alpha_2 x_0 + \alpha_3 + w_0)}$, $b'_0 = (h_0 \prod_{i=1}^{\ell} g_i^{x_i})^{(\alpha_1\alpha_2 x_0 + \alpha_1\alpha_3 + \alpha_1 w_0)}$
- $r'_0 = c'_0 x_0 + \alpha_2 x_0 + w_0 + \alpha_3$
- $h = (h_0 \prod_{i=1}^{\ell} g_i^{x_i})^{\alpha_1}$, $z' = (h_0 \prod_{i=1}^{\ell} g_i^{x_i})^{(x_0\alpha_1)}$

It is then straightforward to check that the following two equalities hold.

$$a'_0 = g_0^{r'_0} h_0^{-c'_0}, \text{ and } b'_0 = h^{r'_0} (z')^{-c'_0}$$

As specified in Figure 3, combining the above equalities leads the user $\mathcal{U}$ to accept the credential as valid.

Wallet-assisted Credential Showing with Inflow and Outflow Prevention. The user holds a credential, with two of its attributes known only to the wallet observer $\mathcal{O}$, and others known only to the user-controlled computer $\mathcal{U}$. As a result, the credential showing requires a special protocol where both $\mathcal{U}$ and $\mathcal{O}$ need to cooperate. This protocol is illustrated in Figure 4. To prevent the establishment of a covert channel between the wallet observer and the verifier, the protocol in Figure 4 has built-in mechanisms for inflow and outflow prevention.

The protocol in Figure 4 highlights a simple setting, where users are only required to prove knowledge of the attributes underlying their credentials. There could be scenarios however, where users are required to prove more elaborate predicates about their attributes. It is still possible to handle those scenarios, since the WBr system [Bra00, Sec. 3.6] allows for proving a wide class of predicates on the attributes. Because of space limitations, we do not discuss general predicate proofs[3] here, and leave them to the full version of the paper.

Correctness of the showing protocol (sketch). To establish the correctness of the showing protocol, we prove that the following equality holds:

[3] For a small example, let us consider a credential h with attributes x_i, $1 \leq i \leq \ell$, and the predicate $\mathcal{P}$: $3x_1 - x_2 = 0$. As shown in Fig. 4, h can be written as $h = h_0^{s_0} \prod_{i=0}^{\ell} g_i^{s_i}$, for $s_i = \alpha_1 x_i$, $1 \leq i \leq \ell$, and $s_0 = \alpha_1$. If we take into account predicate $\mathcal{P}$, then we get $h = h_0^{s_0} (g_1 g_2^3)^{s_1} \prod_{i=3}^{\ell} g_i^{s_i}$. To prove that the x_i's satisfy $\mathcal{P}$, we just need to prove *knowledge* of a discrete log representation of h wrt. basis $(h_0, g_1 g_2^3, g_3, \cdots, g_\ell)$. This can be done by using the same technique of Fig. 4.

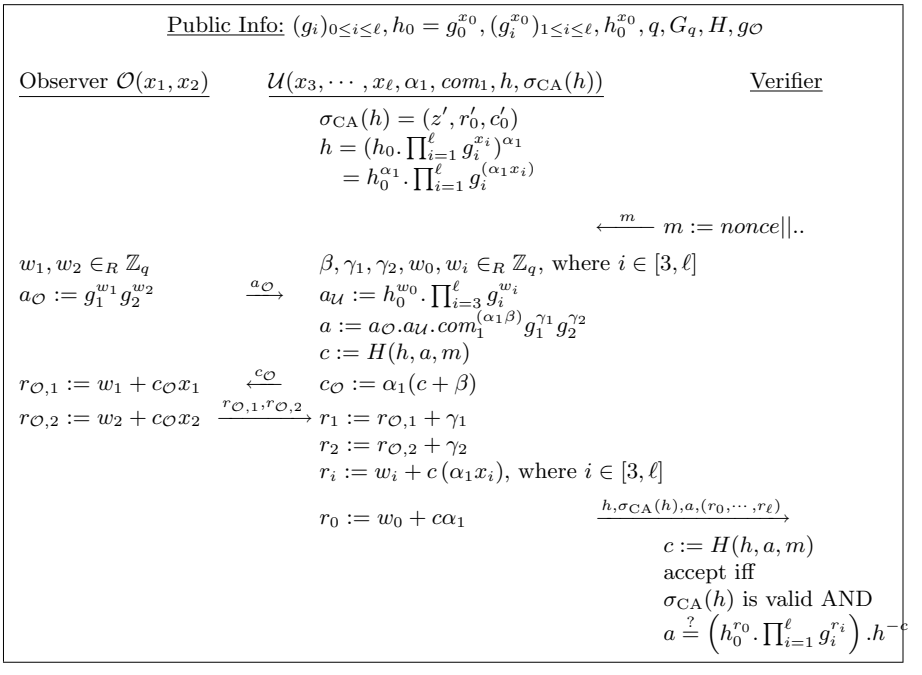


Fig. 4. Wallet-assisted Credential Showing with Inflow and Outflow Prevention (The output tuple $(a, r_0, \cdots, r_\ell)$ represents a signature on the Verifier's message m, using the pair $(\mathcal{O}, \mathcal{U})$'s credential $(h, \sigma_{\mathrm{CA}}(h))$. The protocol can also be seen as a signed proof of knowledge of a discrete log representation of h with respect to basis $(h_0, g_1, \cdots, g_\ell)$).

$$a = \left(h_0^{r_0} . \prod_{i=1}^{\ell} g_i^{r_i}\right) . h^{-c}$$

Starting with the right hand side of the equation we obtain:

$$\begin{aligned}
(\prod_{i=1}^{\ell} g_i^{r_i}) h_0^{r_0} h^{-c} &= (\prod_{i=3}^{\ell} g_i^{w_i + c\alpha_1 x_i}) . g_1^{w_1 + \gamma_1 + (c+\beta)\alpha_1 x_1} . g_2^{w_2 + \gamma_2 + (c+\beta)\alpha_1 x_2} \\
&\quad . h_0^{w_0 + c\alpha_1} . (h_0 \prod_{i=1}^{\ell} g_i^{x_i})^{-\alpha_1 c} \\
&= (\prod_{i=3}^{\ell} g_i^{w_i}) . g_1^{w_1 + \gamma_1 + \beta\alpha_1 x_1} . g_2^{w_2 + \gamma_2 + \beta\alpha_1 x_2} . h_0^{w_0} \\
&= a_{\mathcal{O}} . a_{\mathcal{U}} . (g_1^{\gamma_1} g_2^{\gamma_2}) . (g_1^{x_1} g_2^{x_2})^{\alpha_1 \beta} \\
&= a_{\mathcal{O}} . a_{\mathcal{U}} . (g_1^{\gamma_1} g_2^{\gamma_2}) . com_1^{\alpha_1 \beta} \\
&= a
\end{aligned}$$

Arguments on Inflow and Outflow Prevention capabilities. Briefly stated, the random number β chosen by the user-controlled computer $\mathcal{U}$, is used to mask the challenge $c = H(h, a, m)$, thereby preventing the verifier from covertly communicating with the observer $\mathcal{O}$ through m. Similarly, random numbers γ_1 and γ_2 prevent the observer $\mathcal{O}$ from sending out covert information to the verifier.

5.2 Data Sanitization

In addition to controlling the disclosure of identity information (which we handle by using anonymous credentials), the patient needs a *data sanitization* procedure to anonymize the health records he sends to the monitoring center. As described in Sec. 2, measurements about the patient's health are collected and aggregated in the form of health records. The patient then releases a portion of these records in *sanitized* form to the telemonitoring service. Data sanitization (e.g., [Swe02, CKV+03]) which aims at anonymizing the records, consists of either removing fields that contain identifying information, or modifying the values of those fields through *generalization* or the *addition of some random noise*.[4] The goal of the sanitization procedure, is to blur the direct link between a given record and its owner. Instead of being mapped back to a single individual, a *sanitized record* can generally be associated with a set of possible owners. The latter is called the *anonymity set*, and the actual owner of the sanitized record is said to be *anonymous within this set*. Data sanitization aims at making the anonymity set as large as possible, while keeping the sanitized data useful for subsequent (e.g., statistical) analysis.

Note. It is worth mentioning here, that for the data sanitization techniques above to work, the patient's master device $\mathcal{M}$ should have an *a priori* knowledge of the overall distribution of health measurements among the population, as well as an approximation of the size of the total dataset. Such information need to be updated regularly, and can be made available by the health authorities managing the hospital, for example.

Note that the data exchange in our setting is pseudonymous, since the public part of the patient's credential's $(h, \sigma_{\mathrm{CA}}(h))$ is revealed for every data submission. The latter serves as a pseudonym. The pseudonimity of data is important here; it makes it possible for the monitoring centre to aggregate health records into medical histories associated with each patient's pseudonym. Because of space limitation, we do not discuss the details of data sanitization any further, and leave this topic to the full version of the paper.

[4] The practice of adding noise from a known probability distribution, is one way to anonymize records while keeping the data useful from a statistical point of view. Anonymization through *generalization* consists of replacing the value of a certain field by a more general representation. For example, a specific value in the "age field" of a record can be replaced by an age interval.

6 Proposed Protocol

In the following, we describe the steps taken by each party involved in the telemonitoring process. Figure 5 depicts the whole process. Let $\mathcal{U}$ and $\mathcal{O}$ denote the patient's computer and smartcard respectively.

1. Initially, at the setup phase, the hospital (or monitoring centre) authority issues a credential to the pair $(\mathcal{U}, \mathcal{O})$. This is done using the issuing protocol described in Figure 3. At the end of this step, the pair $(\mathcal{U}, \mathcal{O})$ obtain a credential $(h, \sigma_{\mathrm{CA}}(h))$ containing a number of attributes $x_1, \cdots, x_\ell$, such that x_1, x_2 are only known to the smartcard $\mathcal{O}$, while $x_3, \cdots, x_\ell$ are known only to the patient—represented by $\mathcal{U}$— and not to $\mathcal{O}$.
2. After gathering health readings from the measurement devices, the master monitoring device (also located at the patient's home) sanitizes the data by removing identifying information.[5] The master device then sends the sanitized data to the patient's computer to be approved and signed.
3. The patient's computer $\mathcal{U}$ checks the received data to make sure it has been properly anonymized. If this is the case, $\mathcal{U}$ initiates the signature process with the patient's smartcard $\mathcal{O}$. The signature process is in fact a signed proof of knowledge, of the attributes underlying credential $(h, \sigma_{\mathrm{CA}}(h))$ initially obtained in step 1. The signed proof is performed on the sanitized data. The computation of the signed proof, requires the collaboration of both the smartcard (which knows x_1, x_2) and the patient's computer (which knows $x_3, \cdots, x_\ell$). The necessity of this cooperation makes the signing a two-factor[6] authentication process. The signing is done using the protocol described in Figure 4, where m is chosen to be a concatenation of (1) a *nonce* chosen by $\mathcal{M}$ and (2) the sanitized data to be signed (sEHR). That is $m := nonce||\mathrm{sEHR}$.
4. Once it receives the signed proof $SPK_{\mathrm{patient}}(\mathrm{sEHR})$, the master device $\mathcal{M}$ checks its correctness, and signs on top of it. $\mathcal{M}$ then sends $\sigma_{\mathcal{M}}(SPK)$ and sEHR to the monitoring centre at the hospital, all encrypted under the latter's public key. In order to prevent the monitoring centre from identifying the patient via the signature $\sigma_{\mathcal{M}}(SPK)$, we can set device $\mathcal{M}$ to use a group signature scheme (e.g., [CvH91, ACJT00]) to compute $\sigma_{\mathcal{M}}(SPK)$, instead of a conventional public key signature.

In the setting above, the patient has a single credential $(h, \sigma_{\mathrm{CA}}(h))$ which he uses to sign all the data disclosed to the hospital. This credential can be viewed as a pseudonym of the patient, and used to link all of his health records. This linkability is essential to the process of building medical dossiers. In cases where linkability (between records of the same patient) is not needed, the patient may use a multi-show credential such as those in [CL02, CL04]. Such a credential can be shown an indefinite number of times without the showings being linkable to each other or to the identity of the credential holder.

[5] The data can be sanitized according to rules chosen by the patient and his doctor.

[6] In order to compute a signature with respect to credential $(h, \sigma_{\mathrm{CA}}(h))$, one needs to know the secrets $x_3, \cdots, x_\ell$ and to hold a smartcard containing x_1, x_2.

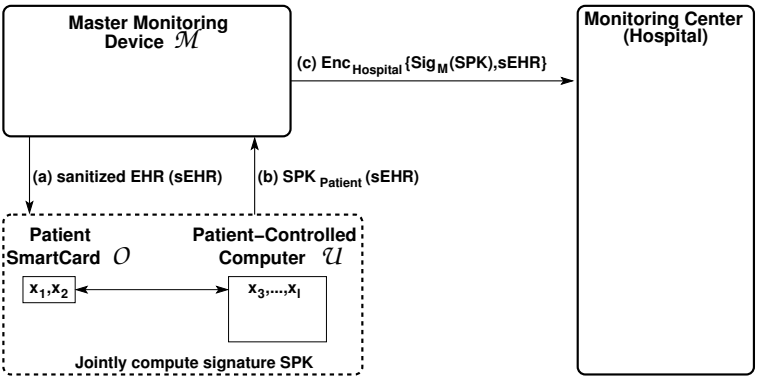


Fig. 5. High-level Protocol Architecture (with two-factor message authentication)

7 Security and Privacy Analysis

In the following, we briefly discuss the security and privacy of our protocol. Our analysis assumes that all the underlying building blocks are secure. A more complete analysis will be given in the full version of the paper.

- **Selective disclosure.** This follows from the *selective disclosure* capabilities of the WBr credential scheme, and the assumption that the tamper-proof device $\mathcal{M}$ will behave according to the protocol specifications.
- **Patient-centricity.** This is achieved due to the fact that it is infeasible to compute a signed proof of knowledge, with respect to a credential, on behalf of its owner. In particular, the presence of a valid signed proof of knowledge (SPK) in the data received by the monitoring centre, indicates that the patient has approved the data release. The data could not have been sent without the patient's approval since (1) no party, except the patient, is able to compute SPK, and (2) $\mathcal{M}$ is trusted to follow the protocol.
- **Pseudonimity and Conditional deanonymization.** This is achieved by the combination of three mechanisms: (1) limiting the disclosure of health data to the sanitized form only, (2) the use, by the patient, of *anonymous credentials* to sign the sanitized data, and (3) the use of a *group signature* scheme by $\mathcal{M}$. The group signature computed by $\mathcal{M}$ convinces the monitoring centre at the hospital, that the signature is generated by a valid master device, without revealing which one. This prevents the monitoring centre from identifying the patient through $\mathcal{M}$. Note however that using the deanonymization mechanism of the group signature scheme, it is possible in case of emergency, to recover the identity of $\mathcal{M}$, and consequently that of the patient.
- **Defense against covert channels.** In the proposed protocol, we use the smartcard-based version of Brands credentials (Fig 4) which has mechanisms to prevent covert inflow and outflow of information between the patient's

smartcard and the monitoring centre at the hospital. This prevents the patient's smartcard from sending out information not approved by the patient.
- **Integrity.** This breaks down into two sub-properties: (1) integrity wrt. alterations caused by the patient, and (2) integrity wrt. alterations occurring during the wire transmission. The first sub-property is satisfied following the soundness[7] of the signed proofs of knowledge, and the assumption that $\mathcal{M}$ will follow the protocol specifications. That is, the fact that $\mathcal{M}$ added its signature on the patient's *SPK*, implies that $\mathcal{M}$ accepted the patient's signature as valid, and found no alteration to the data. The second sub-property is ensured by the unforgeability and soundness of both signature schemes: the group signature and the credential-based signature.
- **Confidentiality.** Assuming that $\mathcal{M}$ is tamper-proof and that it follows the protocol in section 6, the confidentiality property is insured by the fact that all data sent to the hospital is encrypted using a secure encryption scheme.

8 Related Work

The topic of telemedicine has received a lot attention from researchers both in the industry and academia [DMDB07, DDB08, SK08, RLNB03, INT08, AMD08]. Most of the previous works in this area however seem to concentrate mainly on the usability, interoperability, and communication efficiency aspects. The question of protecting the patients' privacy has also been addressed (e.g., [DMDB07, SK08]), but to a lesser extent. In the following we highlight some works from the literature that are most relevant to this paper.

In [DMDB07, DDB08], Durresi et al. propose a ubiquitous health monitoring system which uses existing wireless telecommunication networks to carry data. Owing to the pervasiveness of wireless networks, the system put forward by Durresi et al. makes it possible to send patient data continuously and in real time, to a central medical database. The protocols in [DMDB07, DDB08] address a number of security-related problems such as authentication, confidentiality, and non-repudiation, by taking advantage of the authentication, public-key encryption and signing mechanisms, which come by default with the communication infrastructure (e.g., GSM network.) With respect to privacy, the protocols in [DMDB07, DDB08] achieve only pseudonimity (towards a network observer.) This is done by requiring the patient's cellphone to assume a temporary identity assigned to it by the monitoring centre.

As widely recognized in the literature (e.g., [Bra00, CL02]), pseudonimity *alone* is not sufficient to protect the privacy of patients. In fact, patients' privacy can be guaranteed only if the patient is able to decide (1) what information about his identity is disclosed and to whom, and (2) which portions of his health data can be sent to the monitoring centre. The protocol we propose in this paper satisfies these requirements through the use of *data sanitization* and privacy-preserving credentials, which allow patients to selectively disclose their identity information.

[7] By soundness, we mean that a signed proof that is incorrect, will be detected with overwhelming probability.

In [SK08] Sufi and Khalil propose an efficient data sanitization method for electrocardiograms (ECG). Their method, which includes encoding and compression algorithms, is designed to conceal cardiovascular details, as well as features in ECG data that could identify an individual. The method in [SK08] achieves compression ratios neighbouring 20:1, which makes it suitable for real-time telemonitoring. Our system can use their method as a building block.

Other systems such as [RLNB03, AMD08, INT08, Can08] have dedicated most of their focus to important aspects such as deployability and interoperability, without giving much attention to the privacy issues surrounding telemonitoring systems. This paper addresses medical telemonitoring in a global way, and proposes concrete solutions to protect the privacy of patients.

9 Conclusion

We have presented a protocol for monitoring patients in the comfort of their homes. Our proposed protocol protects patients' privacy at two levels: (1) the *identity information level*: patients are able to selectively reveal information about their identity and to hide the rest, and (2) the *medical data level*: health measurements collected from the patient are sanitized according to patient-approved privacy policies before being sent to the health monitoring centre (*HMC*). The data is sanitized in a way that keeps it useful from a medical perspective, while preventing it from being directly linkable to the patient's identity.

The protocol we propose, provides security against impersonation attacks, even when the patient's computer is compromised. This is achieved thanks to a smartcard-based two-factor authentication mechanism. The same authentication mechanism allows the monitoring center to recognize and accept data records that have been approved for release by the patient, and decline others. Furthermore, and in line with our stated concerns for usability, the procedure we propose for handling disclosure approvals is automated; patients are only required to specify their disclosure policies in an initial configuration phase.

In addition to our ongoing prototype implementation, this work can be extended in a number of ways. For example, in cases where privacy requirements are less stringent, one could use simpler two-factor authentication methods, in particular those based on one-time passwords, generated by a tamper-proof device. We can also improve the communication efficiency of our protocol by using sanitization and compression techniques such as those in [SK08].

The issue of *liability* also deserves further investigation. For example, if a patient dies, the monitoring center should be able to prove that everything that could be done to save the patient has been done. A simple way to achieve liability would be to require the *HMC* to send an acknowledgement token back to the patient's master device, every time it receives data from the patient. These tokens can be used later to prove that the *HMC* was aware of the patient's condition. In addition, the *HMC* should keep a record of all the efforts it made to help the patients (e.g., ambulance calls etc.) All of these records, as well as the

acknowledgement tokens, can constitute the basis of any service audits that may follow. More details on the question of liability will be presented in future work.

Acknowledgements. This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy and the Research Fund K.U.Leuven and by the IWT-SBO project (ADAPID) "Advanced Applications for Electronic Identity Cards in Flanders".

References

[ACJT00] Ateniese, G., Camenisch, J., Joye, M., Tsudik, G.: A practical and provably secure coalition-resistant group signature scheme. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 255–270. Springer, Heidelberg (2000)

[AMD08] AMD Telemedicine Inc. Telemedicine becoming more relevent to childrens health within school systems (December 2008)

[Bra00] Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge (2000)

[Can08] Canada Health Infoway. Patients manage health at home with telehealth (November 2008), `http://www.infoway-inforoute.ca`

[CKV+03] Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.: Tools for privacy preserving distributed data mining. ACM SIGKDD Explorations 4(2) (2003)

[CL02] Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)

[CL04] Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)

[CvH91] Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)

[DDB08] Durresi, A., Durresi, M., Barolli, L.: Secure ubiquitous health monitoring system. In: Takizawa, M., Barolli, L., Enokido, T. (eds.) NBiS 2008. LNCS, vol. 5186, pp. 273–282. Springer, Heidelberg (2008)

[DMDB07] Durresi, A., Merkoci, A., Durresi, M., Barolli, L.: Integrated biomedical system for ubiquitous health monitoring. In: Enokido, T., Barolli, L., Takizawa, M. (eds.) NBiS 2007. LNCS, vol. 4658, pp. 397–405. Springer, Heidelberg (2007)

[Ein08] Einthoven, W.: (2008), `http://nobelprize.org`

[INT08] Intel Digital Health Group Inc. Addressing the challenges of chronic illness with personal health system technology (2008)

[Nag06] Nagy, B.: Telemedicine's depth now going beyond rural areas. Managed Healthcare Executive (2006), `http://www.managedhealthcareexecutive.com`

[Oka92] Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)

[RLNB03] Rialle, V., Lamy, J.-B., Noury, N., Bajolle, L.: Telemonitoring of patients at home: a software agent approach. Computer Methods and Programs in Biomedicine 72(3), 257–268 (2003)

[SK08] Sufi, F., Khalil, I.: Enforcing secured ecg transmission for realtime telemonitoring: A joint encoding, compression, encryption mechanism. Security and Communication Networks 1(5), 389–405 (2008)

[Swe02] Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10(5), 571–588 (2002)

[TSI+08] Trief, P.M., Sandberg, J., Izquierdo, R., Morin, P.C., Shea, S., Brittain, R., Feldhousen, E.B., Weinstock, R.S.: Diabetes management assisted by telemedicine: Patient perspectives. Telemedicine and e-Health 14(7), 647–655 (2008)

[US08] The assisted-living project (November 2008), `http://lion.cs.uiuc.edu/assistedliving`

[Zep08] Zephyr™: Smart Fabric for physiological and bio mechanical monitoring (2008), `http://www.zephyrtech.co.nz/technology`

Analysis of Data Dependency Based Intrusion Detection System*

Yermek Nugmanov[1], Brajendra Panda[1], and Yi Hu[2]

[1] Computer Science and Computer Engineering Department
University of Arkansas
Fayetteville, AR 72701
{ynugmano,bpanda}@uark.edu
[2] Computer Science Department
Northern Kentucky University
Highland Heights, KY 41099
huy1@nku.edu

Abstract. This research focuses on analyzing the cost effectiveness of a database intrusion detection system that uses dependencies among data items to detect malicious transactions. The model suggested in this paper considers three main factors: the quality of intrusion detection, the probability of intrusion, and the cost structure of an organization whose data is protected by the intrusion detection system. We developed a step by step approach that helps in determining the optimal configuration expressed by the response strategy and the threshold value. The experimental results show that our model is capable of finding the optimal configuration while taking the cost structure of an organization into consideration.

Keywords: Database intrusion detection, semantic analyzer, cost analysis, response strategy.

1 Introduction

Although there exist many security tools for protecting computer systems from attacks, as per [1], none of them can provide absolute security. Therefore, all important events that occur in a computer system must be supervised and examined for a possible presence of malicious activity by intrusion detection systems. Of the two universally recognized models for intrusion detection [2, 14], misuse detection and anomaly detection, the latter typically uses a threshold to define which events are considered normal and which are considered intrusive [3, 4, 15]. By changing the threshold, an organization may find the optimal balance between successful detections and false alarms. However, finding of the optimal configuration is a difficult task. Some recent works have focused on the optimization techniques for anomaly detection systems

* Research of Yermek Nugmanov and Brajendra Panda was supported in part by AFOSR under grant FA 9550-04-1-0429. Research of Yi Hu was supported in part by the KY NSF EPSCoR program.

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 111–125, 2009.

[4, 5]. At present, the majority of existing host-based anomaly detection systems are ineffective in detecting attacks on databases, since they are focused on tracking and analyzing events that occur in operating systems and applications, and not on the database itself. Although a few models have been developed for detecting malicious activities in databases, to the best of our knowledge, none of these methods have been analyzed for their cost–effectiveness. The objective of this work is to evaluate the data dependency based database intrusion detection system [12] for optimization based on response strategy and threshold value.

2 Background

A limited research has been done to address the problem of malicious transaction detection in database systems. In [6] an architecture for intrusion-tolerant database systems is proposed. An intrusion-tolerant database management system is able to operate and deliver essential services even in case of attacks. However, this approach is more focused on the localization of attacks and recovery of the damage, than on developing a specific intrusion detection system. A database intrusion detection scheme based on data dependency rule mining is presented in [11]. A similar method that uses weighted sequence mining techniques is offered in [7]. The major drawback of this detection method is that the weights of attributes must be assigned manually. The method presented in [8] detects intrusion in databases that employs role-based access control and it uses Naïve Bayes Classifier to predict the role which the observed SQL command most likely belongs to, and compares it with the actual role. If the roles are different, the SQL statement is considered illegal. An intrusion detection system for real-time database systems has been discussed in [9]. Researchers in [10] proposed a misuse detection system for databases based on the observation that there exist certain regularities in "access patterns" of users. Our research is based on the model presented in [12] that detects malicious activities in a database management system by using data dependency relationships.

Regarding the effectiveness of intrusion detection, a study presented in [3] showed that such methods are subject to the base-rate fallacy, coming to the conclusion that "in order to achieve substantial values of the Bayesian detection rate, we have to achieve a low false alarm rate". He found that in most cases such a rate is unattainable. Following that, researchers in [13] developed the techniques for building an intrusion detection system on the basis of cost-sensitive models. The first comprehensive study on the cost effectiveness of intrusion detection systems appeared in [5], which addresses the problem of finding the optimal configuration of a single intrusion detection system, and various combinations of multiple intrusion detection systems. An optimization scheme based on game theory has been offered in [4].

3 The Model

3.1 Data Dependency Based Intrusion Detection Model

In this section, we very briefly discuss the data dependency based database intrusion detection model, which was presented in [12]. This model has two components,

namely, the static semantic analyzer and the dynamic semantic analyzer. The static semantic analyzer is employed to analyze the database application program statically to discover intra-transaction data dependencies represented by the read, pre-write, and post-write set. If a transaction does not conform to the read, pre-write, or post-write sets, it will be identified as a malicious transaction. This is treated as the first line of defense. In case, the malicious transactions are well-crafted and are compliant with the data dependencies discovered by the static semantic analyzer, the access patterns to these sets can be used to discover malicious transactions. The access probabilities for these sets depend on the execution path of the database application and the normal user access patterns of the database. The dynamic semantic analyzer is designed to calculate the access probability based on the database log.

To have a better understanding on the data dependency based database intrusion detection model, we illustrate an example here. Suppose during normal database operation phase, two transactions T_1 and T_2 are generated by the database application. Assume that we have the SQL statements in T_1 and T_2 as shown in Table 1.

Table 1.

T_1	T_2
Update Table1 set m = i + j where … Update Table1 set n = m + k where … Update Table1 set t = m + p + q where …	Update Table1 set m = i + r where…

The static semantic analyzer will generate the read, pre-write, and post-write sets as illustrated in Table 2. Let us use data item *m* to illustrate the purpose of these sets and how they can be used to identify malicious database transaction. Data item *m* has non-empty read set and post-write set. The read set states that before data item *m* is updated by the transaction, either data items in {*i, j*} or {*i, r*} have to be read by the transaction. The post-write set states that after *m* is updated, data item *n* and *t* have to be updated by the same transaction. Please note that the where clauses have been ignored to keep this example short and may not be so in reality. If a transaction updates data item *m* without complying with rules specified by the read set and post-write set, it will be identified as an anomalous transaction.

Table 2.

	Read Set	Pre-Write Set	Post-Write Set
m	{ {i, j}, {i, r}	{ ∅ }	{ {n,t}}
n	{ {m, k} }	{ {m} }	{ ∅ }
t	{ {m, p, q} }	{ {m, n} }	{ ∅ }

If an attacker's transaction is well-crafted and conform to the data dependencies specified, the dynamic semantic analyzer then steps in. Say, in reality, the access probabilities of {*i, j*} and {*i, r*} in the read set of *m* are different for normal user

transactions and set $\{i, r\}$ is only infrequently used for updating m in special occasions. If the attacker's transaction updating m reads $\{i, r\}$ instead of $\{i, j\}$ before modifying m, the access probability of the read set generated by the dynamic semantic analyzer can be used to identify this anomalous transaction. For more information on the data dependency based database intrusion detection model, interested readers may refer to [12].

As per the requirement of the data dependency based database intrusion detection method, there can be only two types of users: normal users and intruders. Normal users are authorized users, who connect to the database only through the database application and, therefore, can generate only legal transactions. Intruders are unauthorized users, who connect to the database from a remote terminal masquerading as normal users. There are many ways to pretend to be a normal user. For example, an intruder can obtain a password of the legitimate account or take control of a normal user's database connection. In any case, in order to get an access to the database, an intruder must find some vulnerability and exploit it.

3.2 Probability of Intrusion

There are three main factors that contribute to the cost of anomaly detection. These factors should be taken into consideration in the analysis of the optimal configuration of a data dependency based intrusion detection system. The first factor is the detection rate of the system. The method described in [4] was adapted to estimate this parameter. The second factor is the ratio of intrusions to legal activities. We assess this value by assuming that the number of intrusions depends on the number of vulnerabilities in the computer system that may allow an attacker to establish a connection to the database protected by the system. Bayes' formula for posterior probability is used to find the actual probability of attack in presence or absence of an alarm signal. The last factor is the losses incurred by an organization in different outcomes of a single attack. These values define which response strategy the organization needs to apply. The optimization conditions for the response strategy are found by means of linear programming. The optimal value of threshold is derived computationally.

If both valid and illegal transactions are generated at the same rate, the expected rate of intrusions among all transactions can be expressed the as follows:

$$\lambda = \frac{\alpha\psi}{\alpha\psi + \varphi}$$

where α is the proportion of successfully exploited vulnerabilities to all discovered vulnerabilities, ψ is the rate at which vulnerabilities are discovered and φ is the rate at which normal users establish connections to the database. The value of φ can be easily retrieved by analyzing the database log. In order to find ψ, it is necessary to find out what sort of vulnerabilities can be used by an intruder to establish a connection to the database. For example, if the operating system on which the database application is deployed has a vulnerability that allows getting unauthorized access to this system, the intruder finally will be able to steal the password of a legitimate database user and establish a connection to the database. Then, using the information supplied by the vendor or other organizations such as Computer Emergency Response

Team, we can estimate the rate at which such vulnerabilities are discovered. The most difficult is to evaluate the value of α which depends on a large number of various factors.

The probability that certain vulnerability will be successfully used for an attack is greatly influenced by the personality of the individual who first discovered this vulnerability. If the vulnerability was discovered by the vendor, it will very likely not be announced before the hot fix is released. If the vulnerability is discovered by a potential intruder, everything depends on the intruder's behavior: a hacker can either conduct an attack or publish the discovered vulnerability. We can assume that, in the first case, there is always a possibility that instead of attacking a single target, the hacker can randomly choose a target or perform a mass attack against all known users of the vulnerable software or hardware. If the vulnerability is published, the probability that this vulnerability will be used against a single organization considerably increases. All these factors significantly complicates the calculation of α. Hence, to simplify the problem, we assume that α is the fraction of vulnerabilities successfully utilized by hackers before the patch has been applied by a vendor. Thus, relying on the statistics of successful attacks we can estimate the value of α, which, in turn, allows us to find the intrusion rate.

3.3 Evaluation of the Data Dependency Based Database Intrusion Detection

As explained earlier, the data dependency based intrusion detection system uses the dependencies among the data items in the database. Before a data item is updated in the database, some other data items are read or written, and after the update, other data items can be written too. Malicious transactions are detected by comparing read, pre-write and post-write sets against data items actually read or written by user transactions. Also the access probability for these sets can be used to identify malicious transactions. Without loss of generality we can formulate data dependency based intrusion detection method as the following two rules:

1. *Static Detection Rule*: Each update operation of a transaction must conform to the data dependencies represented by the read, pre-write, and post-write sets;
2. *Dynamic Detection Rule*: Each update operation of a transaction must conform to the normal user access patterns of different sets in the read, pre-write, and post-write sets.

Normally the update operations of a transaction are sequentially tested for compliance with both the rules. The transaction is considered illegal if any of its update operations does not conform to at least one of the rules. However, for the purposes of analysis we assume that initially all of the update operations of a transaction are tested for compliance with the first rule. We call this procedure static detection. Then, the update operations are tested for compliance with the second rule, which we call dynamic detection. The transaction is considered illegal if any of its update operations fails to pass either the static or the dynamic detection procedure. It is obvious that our approach produces the identical results with the original detection method.

Each transaction that goes through the static analyzer causes the creation of the data dependency which the transaction conforms to. In other words, transactions that were used to generate the data dependencies are always identified as legal by the

static detection. Since, by definition, the static analyzer takes into consideration all transactions that can be generated by the database application, we can assume that the false positive rate of the static detection equals to zero. Actually, this assumption seems to hold in most real-life situations, as the transactions which were omitted during the static analysis are likely to be revealed later, in course of generating the log file for dynamic analysis. The experiments conducted by [12] provide the empirical evidence of this assumption, since, according to their results, the false positive rate always equals to zero, when the detection is based only on the results of static semantic analysis.

Notice that the transactions identified as malicious by the static detection will be classified as illegal regardless of the result of the dynamic detection. Therefore, if we denote P_{STP} as the true positive rate of the static detection, then the aggregate detection rate of both static and dynamic detection can be calculated as follows:

$$P_{TP} = P_{STP} + (1 - P_{STP}) P_{DTP}$$

where P_{DTP} is the rate at which the dynamic detection correctly identifies intrusions misclassified by the static detection.

The static detection cannot be made more or less strict, so that P_{STP} is a constant value unless the database application is modified. P_{STP} can be expressed as the proportion of the transactions identified as illegal by the static detection to all illegal transactions. To obtain this proportion, we need to construct a set of sample intrusions and test the static detection procedure against this set. If the selected samples are representative, we will get an indicative P_{STP} value.

Unlike the static detection, the dynamic detection can be made more or less strict by regulating the threshold value. This means that there exists a Receiver Operating Characteristic (ROC) curve which plots P_{DTP} against false positive rate P_{FP}. Although the ROC curve can also be derived empirically, we will need to conduct a large amount of tests. Therefore, it is better to use an analytical method for finding economically optimal configuration of dynamic detection.

Normally, the dynamic analyzer computes the total use probability for each data item set and marks it as infrequently used if the value is less than the threshold τ. In order to comply with data dependencies, a transaction needs to access all data items of at least one data item set of each read, pre-write and post-write sets that represent these data dependencies. The dynamic detection generates an alarm signal when the transaction includes at least one data item set which is marked as infrequently used. Let us make some alterations of this method. Suppose, the data dependencies created by the static analyzer altogether contain N data item sets, D_1, D_2,…,D_N. Instead of marking data item sets as infrequently used, the dynamic analyzer associates each data item set with its total use probability M_{D_i}, $1 \le i \le N$. Now, suppose that transaction T accesses all members of K different data item sets, $K \le N$. During the analysis of T, the dynamic detection finds the minimum μ of the values associated with each of K data item sets,

$$\mu = \min\{M_{D_1}, M_{D_2}, ..., M_{D_K}\}.$$

Then, this value is compared against the threshold τ. If μ is greater than or equal to τ, then transaction *T* does not include any infrequently used data item, i.e., all *K* data items have the total use probability that are greater than the threshold and the transaction is normal. If μ is less than τ, then the transaction includes at least one infrequently used data item, so that an alarm signal is generated. Even though the alterations we made do not change the outcome of the dynamic detection, they rearrange this procedure to the form to which we can apply an analytical method. The only exception is that the result is inverted with regard to the threshold value, so that we need to change the limits of integration. As a result, using the base formulas from [17], we express P_{DTP} and P_{FP} as follows:

$$P_{DTP} = \int_{-\infty}^{\frac{\tau-\mu_F}{\sigma_F}} U(x)dx = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{\tau-\mu_F}{\sigma_F}} e^{-\frac{x^2}{2}} dx,$$

$$P_{FP} = \int_{-\infty}^{\frac{\tau-\mu_L}{\sigma_L}} U(x)dx = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{\tau-\mu_L}{\sigma_L}} e^{-\frac{x^2}{2}} dx;$$

where τ is the detection threshold, μ_L is the mean value of μ for legal transactions, μ_F is the mean value of μ for intrusions, σ_L is the variance of μ for legal transactions, σ_F is the variance of μ for intrusions, and *U(x)* is the probability density function for both normal and malicious database transactions. Same variance for normal and malicious database transaction is assumed. So essentially, P_{DTP} or P_{FP} represents the integration of the normal density function. Figure 1 illustrates a sample computation of P_{DTP} and P_{FP}.

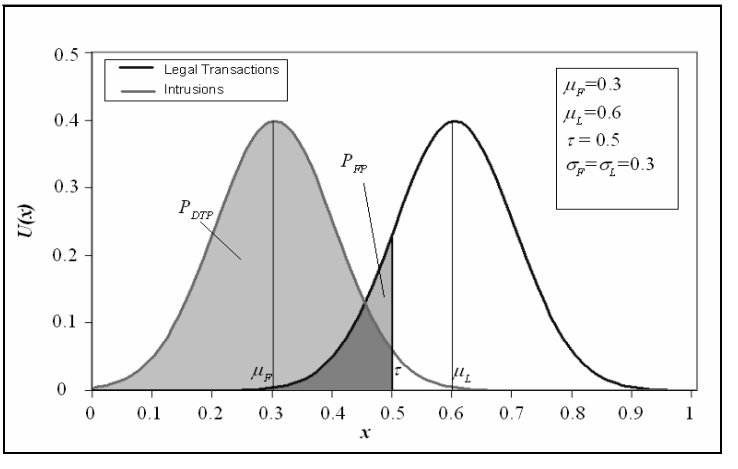


Fig. 1. Computation of P_{DTP} and P_{FP}

Before performing analytical analysis we need to estimate the values of μ_L, and σ_L, μ_F and σ_F. The former two values can easily be derived from the log file. In order to obtain the latter two values we need to find illegal transactions that are classified as legal by the static detection and build a set of sample intrusions. The samples must be representative; otherwise, the computed value will not indicate the real false positive rate.

3.4 Optimal Response Strategy

An organization may or may not have to respond to the alarm signal generated by the intrusion detection system. The response usually consists of manual investigation of the event which caused the alarm. If the event is indeed malicious in nature, the manual investigation allows recovering a part of the damage caused by the intrusion. However, the investigations are costly, since they engage resources, both people and equipment, and often interfere with the ongoing work. Therefore, an organization must decide to investigate an alarm signal, only if there is a great likelihood that the alarm was caused by an intrusion.

In order to make a reasoned decision, an organization needs to find the probability of intrusion given occurrence of alarm, which is expressed, in case of data dependency based intrusion detection, by the following formula:

$$p_A = P(I \mid A) = \frac{P(A \mid I)P(I)}{P(A \mid I)P(I) + P(A \mid \neg I)P(\neg I)} = \frac{P_{TP}\lambda}{P_{TP}\lambda + P_{FP}(1-\lambda)}$$

where λ is the intrusion rate. In fact, an organization may decide to launch an investigation even if there is no alarm signal, assuming that the intrusion detection system produced a false negative outcome. In this case, the organization will need to evaluate reliability of its intrusion detection system by computing the posterior probability of intrusion given the absence of an alarm signal, as follows:

$$p_N = P(I \mid \neg A) = \frac{P(\neg A \mid I)P(I)}{P(\neg A \mid I)P(I) + P(\neg A \mid \neg I)P(\neg I)} = \frac{(1-P_{TP})\lambda}{(1-P_{TP})\lambda + (1-P_{FP})(1-\lambda)}$$

If an investigation finds that a suspicious transaction in fact is illegal, the organization starts recovery procedure to restore the data damaged by this transaction.

Since investigations are costly, an organization may skip the investigation procedure and immediately start the database damage assessment and recovery procedures to cancel out the effects of the transaction which caused an alarm signal. That is, the organization fully relies on the decision made by the intrusion detection system. However, in case of detection error, the organization will incur losses by rolling back a legal transaction. In contrast, a manual investigation always clarifies the true type of a transaction, so that the recovery that follows a manual investigation involves only illegal transactions.

Let c_i denote the cost of manual investigation, c_r represent the cost of a rollback operation, c_d represent the damage caused by a successful intrusion, and c_e denote the loss caused by a rollback operation over a legal transaction. Then, the expected cost of a transaction when an alarm is generated is determined by the following equation:

$$F_A(i_A, r_A) = c_i i_A + p_A c_r i_A + c_r r_A + p_A c_d (1 - i_A - r_A) + (1 - p_A) c_e r_A =$$
$$= (c_i + p_A c_r - p_A c_d) i_A + (c_r + c_e - p_A c_d - p_A c_e) r_A + p_A c_d;$$

where i_A is the fraction of manually investigated alarm signals, and r_A is the rate at which the transactions identified as illegal are automatically rolled back by employing a database recovery procedure. This function, however, does not reflect the entire expected cost, as it does not take into consideration the losses incurred in the absence of the alarm, which are computed as follows:

$$F_N(i_N, r_N) = c_i i_N + p_N c_r i_N + c_r r_N + p_N c_d (1 - i_N - r_N) + (1 - p_N) c_e r_N =$$
$$= (c_i + p_N c_r - p_N c_d) i_N + (c_r + c_e - p_N c_d - p_N c_e) r_N + p_N c_d;$$

where i_N is the rate at which transactions classified as legal are investigated, and r_N is the rate at which transactions classified as legal are automatically rolled back. Figure 2 illustrates the possible responses in both alarm and no alarm cases.

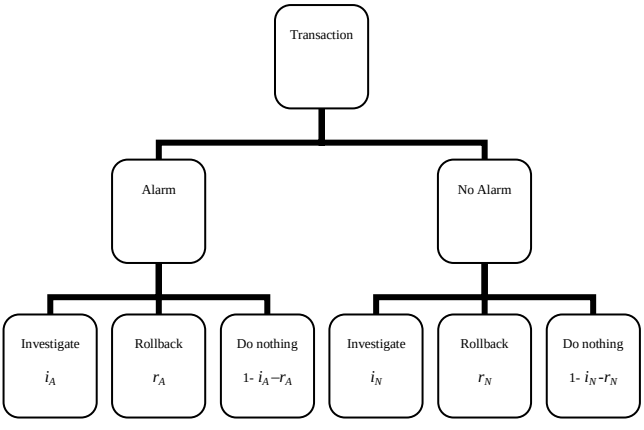


Fig. 2. Possible Responses

From the expected costs of a transaction in both alarm and no alarm cases, we can determine the total cost as the arithmetic mean of these values:

$$F_T = P(A) \times F_A + P(\neg A) \times F_N =$$
$$= (P_{TP}\lambda + P_{FP}(1-\lambda)) \times F_A + ((1-P_{TP})\lambda + (1-P_{FP})(1-\lambda)) \times F_N.$$

The variables i_A, r_A, i_N, r_N define the optimal response strategy and must be chosen to provide minimal values of F_A and F_N. The optimization conditions with regard to these functions are identical, since both of them are expressed by the same formula; the only exception is the different probability parameters. We examine the optimization conditions by the example of F_A.

The task of finding the minimal cost can be represented as the problem of linear programming [16] with an objective function $F_A(i_A , r_A)$ that is a subject to the following inequality constraints:

$$0 \le r_A \le 1,\ 0 \le i_A \le 1, \text{ and } 0 \le i_A + r_A \le 1.$$

These inequalities follow from the fact that i_A and r_A represent the proportions of responses to the received alarm signals. As shown in Figure 3, they produce the region of possible solutions limited by both coordinate axes and the line expressed by function $i_A(r_A) = 1 - r_A$.

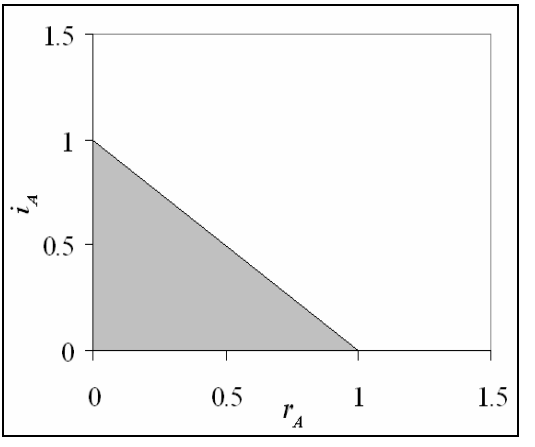


Fig. 3. Solution Region for F_A

It is known that at least one of the vertices of feasible region represents the optimal solution [16]. In our case, there exist three possible solutions that correspond to the vertices of the triangle-shaped region:

1. i_A=0, r_A=0; 2. i_A=1, r_A=0; 3. i_A=0, r_A=1.

Each of these solutions becomes optimal under certain conditions. These conditions are defined by the coefficients of the variables i_A and r_A. For simplicity, we represent these coefficients as follows:

$$a = c_i + p_A c_r - p_A c_d,\ \ b = c_r + c_e - p_A c_d - p_A c_e .$$

In this case, we can represent F_A as

$$F_A(i_A, r_A) = a i_A + b r_A + p_A c_d .$$

It is obvious that i_A=0 and r_A=0 is the optimal solution if $a \ge 0, b \ge 0$.

We can prove it by contradiction. First, let us find the minimal value of the expected cost:

$$F_A(0,0) = a \times 0 + b \times 0 + p_A c_d = p_A c_d .$$

Now, let us assume that there exist such i_A and r_A that $F_A(i_A, r_A)$ is less than $p_A c_d$, i.e.

$$ai_A + br_A + p_A c_d < p_A c_d .$$

In this case, the following condition is necessary to hold:

$$ai_A + br_A < 0 .$$

Since the inequality constraints prevent i_A and r_A from accepting negative values, this condition contradicts to the initial condition that both a and b are greater than or equal to zero.

i_A=1 and r_A=0 is the optimal solution if $a < 0$, $a \le b$.

In this case the minimal value for $F_A(i_A, r_A)$ is

$$F_A(1,0) = a \times 1 + b \times 0 + p_A c_d = a + p_A c_d .$$

Let us assume there exist such i_A and r_A, $r_A > 0$, $i_A \le 1 - r_A$, that $F_A(i_A, r_A)$ accepts a lesser value, i.e.,

$$ai_A + br_A + p_A c_d < a + p_A c_d .$$

Knowing that

$$ai_A \le a(1 - r_A),$$

we can represent the previous inequality in the form of

$$a - ar_A + br_A + p_A c_d < a + p_A c_d .$$

Simplifying it, we will get $a > b$, which contradicts to $a \le b$.

i_A=0 and r_A=1 is the optimal solution if $b < 0$, $b < a$.

We can prove this condition in the same way as the previous one.

Now we can calculate what ratio of the real values each optimal solution corresponds to. From $a \ge 0$, follows that

$$c_i + p_A c_r - p_A c_d \ge 0,$$

i.e.,

$$\frac{c_i}{c_d - c_r} \ge p_A .$$

From $b \ge 0$, follows that

$$c_r + c_e - p_A c_d - p_A c_e \ge 0,$$

i.e.,

$$\frac{c_r + c_e}{c_d + c_e} \geq p_A .$$

From $a \leq b$, follows that

$$c_i + p_A c_r - p_A c_d \leq c_r + c_e - p_A c_d - p_A c_e ,$$

i.e.,

$$\frac{c_r + c_e - c_i}{c_r + c_e} \geq p_A .$$

Thus, the optimization conditions for F_A can be formulated as shown in Table 3.

Table 3.

Optimization conditions	i_A	r_A
$\frac{c_i}{c_d - c_r} \geq p_A$ and $\frac{c_r + c_e}{c_d + c_e} \geq p_A$	0	0
$\frac{c_i}{c_d - c_r} < p_A$ and $\frac{c_r + c_e - c_i}{c_r + c_e} \geq p_A$	1	0
$\frac{c_r + c_e}{c_d + c_e} < p_A$ and $\frac{c_r + c_e - c_i}{c_r + c_e} < p_A$	0	1

The same conditions for F_N are achieved by replacing P_A with P_N.

To sum up, the organization's actions to optimize the transactions should consist of the following steps:

1. Select a representative set of known illegal transactions and test the static detection against the set to estimate P_{STP} as the proportion of the transaction identified as intrusions to all transactions in the set;
2. Find μ for each of the illegal transactions identified as legal by the static detection, compute mean μ_F and variance σ_F;
3. Select the representative set of known legal transactions and find μ for each of these transactions; compute mean μ_L and variance σ_L.
4. Determine α, ψ and φ and calculate the rate of intrusions λ;
5. Determine the organization's cost metrics expressed by c_i, c_r, c_e, c_d ;
6. Find the optimal configuration, i.e. the values of F_T, τ, i_A, r_A , i_N, r_N;
7. Set the threshold to τ;
8. Follow the response strategy defined by i_A, r_A , i_N, r_N .

Since many of the initial variables change their values in course of time, the organization must periodically repeat these operations to update the optimal configuration.

4 Experimental Results

As mentioned in Section 3.2, there are three factors affecting the cost of the intrusion detection. Changing of any of these factors leads to the alteration of the final outcome. However, as Table 3 indicates, the absolute values of the cost metrics expressed by c_i, c_e, c_r and c_d are less important than the proportions of these four variables. The proportions of the first two parameters are unlikely to change with time; moreover, the values of c_i and c_e are expected to be of the same magnitude, as both the investigation of an alarm signal and the recovery from an erroneous rollback require human interaction. In contrast, the value of c_d may significantly vary, as the damage caused by an attack depends on the nature of the data stored in the database. Given that the rollback is one of the key features of database management systems, c_r is expected to be much smaller than c_i and c_e. From these considerations, the following variants of the cost metrics were selected for testing:

1. $c_r = 1$, $c_e = c_i = 100$, $c_d = 10000$;
2. $c_r = 1$, $c_e = c_i = 100$, $c_d = 50000$;
3. $c_r = 5$, $c_e = c_i = 100$, $c_d = 50000$;

In the first variant, we assume that the damage caused by a successful intrusion is 100 times greater than the cost of investigation and the cost of recovery from a falsely conducted rollback, which, in turn, are 100 times greater than the cost of a rollback operation. In the second variant we increase the potential damage, while other parameters remain the same. In the third variant we additionally increase the cost of a rollback operation.

The quality of intrusion detection is expressed by P_{STP}, μ_L, μ_F ,σ_L and σ_F. These parameters should not significantly change with time, unless the intrusion detection system has been deployed without a proper training. We tested the program against different sets of the parameters, and selected those whose test results are more illustrative: $P_{STP} = 0.4$, $\mu_L=0.6$, $\mu_F =0.4$, $\sigma_L=0.2$, $\sigma_F=0.2$. In order to show how the probability of intrusion influences the optimal configuration, we chose the value of λ to vary from 10^{-5} to 10^{-1}. Figure 4 illustrates the total expected cost for all three variants of cost metrics. The x-axis of the graph is a logarithmic scale representing the values of λ. It must be noted that the data provided in this figure is a small subset of that obtained through the experiment; due to page limitation, we could not produce the entire result here.

As Figure 4 indicates, in spite of the fact that in the second variant, the value of potential damage c_d is five times greater than that in the first one, the maximum value of the cost function does not increase significantly. However, a larger value of c_d increases the growth rate of the cost function. In contrast, the cost of recovery from an erroneous rollback c_r affects the maximum value of the cost function. Furthermore, as Figure 4 depicts, the costs start to decline at some point. Although it seems atypical, the explanation for this phenomenon is that the growth of intrusion rate decreases

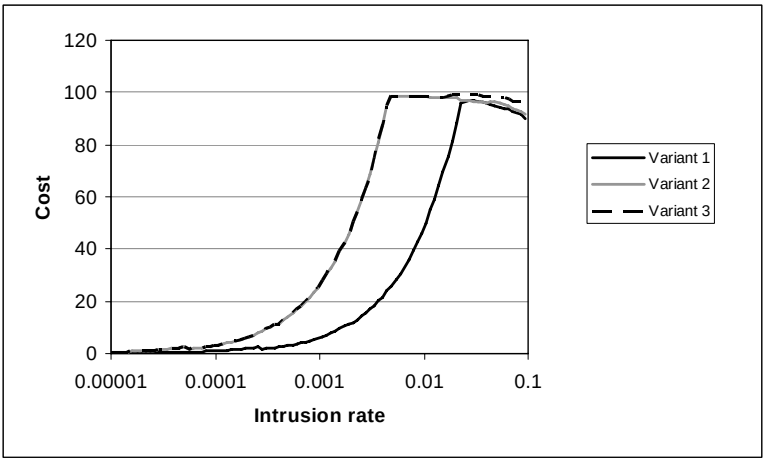


Fig. 4. Total Cost for Different Variants of Cost Metrics

the uncertainty of intrusion detection process. In other words, high intrusion rate permits more accurate decisions, thus, effectively responding to the attacks.

5 Conclusions

In this paper, we have presented a model that can be used to determine the economically optimal configuration of the data dependency based intrusion detection system. Three main parameters are taken into consideration in our model: the intrusion rate, the quality of the intrusion detection, and the cost metrics of an organization. We presented a step by step methodology that helps in finding the optimal configuration, expressed by the response strategy and the threshold value. Our experimental results suggest that the value of potential damage does not proportionately affect the total cost function. However, the recovery cost associated with an erroneous rollback significantly affects the maximum value of the cost function. Furthermore, the experiment illustrated that the dynamic detection is useful only at high intrusion rates. Therefore, while the expected rate of intrusion remains reasonably low, an organization may simply rely on the static detection and still maintain a low total expected cost of the intrusion detection. As part of our future work, we plan to conduct further experiments to identify optimal response strategy under various circumstances.

Acknowledgement

This research has been supported in part by US AFOSR under grant FA 9550-04-1-0429 and the KY NSF EPSCoR program. We are thankful to Dr. Robert. L. Herklotz and Jeffrey Mossey for their support, which made this work possible.

References

1. Richardson, R.: 2007 CSI Computer Crime and Security Survey, Computer Security Institute (2007), http://gocsi.com
2. Axelsson, S.: Intrusion Detection Systems: A Survey and Taxonomy, Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden (2000), http://www.cs.chalmers.se/~sax/pub/
3. Axelsson, S.: The Base-rate Fallacy and the Difficulty of Intrusion Detection. ACM Transactions on Information and System Security 3(3), 186–205 (2000)
4. Cavusoglu, H., Misra, B., Raghunathan, S.: Optimal Configuration of Intrusion Detection Systems. In: Proc. Second Secure Knowledge Management Workshop, pp. 1–7 (2006)
5. Ulvila, J.W., Gaffney, J.E.: Evaluation of Intrusion Detection Systems. Journal of Research of the National Institute of Standards and Technology 108(6), 453–473 (2003)
6. Liu, P.: Architectures for Intrusion Tolerant Database Systems. In: Proc. 18th Annual Computer Security Applications Conference, pp. 311–320 (2002)
7. Srivastava, A., Sural, S., Majumdar, A.K.: Database Intrusion Detection using Weighted Sequence Mining. Journal of Computers 1(4), 8–17 (2006)
8. Bertino, E., Kamra, A., Terzi, E., Vakali, A.: Intrusion Detection in RBAC-administered Databases. In: Proc. 21st Annual Computer Security Applications Conference, pp. 170–182 (2005)
9. Lee, V., Stankovic, J., Son, S.: Intrusion Detection in Real-time Databases via Time Signatures. In: Proc. Sixth IEEE Real-Time Technology and Applications Symposium, pp. 124–133 (2000)
10. Chung, C., Gertz, M., Levitt, K.: DEMIDS: A Misuse Detection System for Database Systems. In: Proc. Integrity and Internal Control in Information Systems: Strategic Views on the Need for Control, IFIP TC11 WG11.5, Third Working Conference, pp. 159–178 (2000)
11. Hu, Y., Panda, B.: A Data Mining Approach for Database Intrusion Detection. In: Proceedings of the 19th ACM Symposium on Applied Computing, Nicosia, Cyprus (2004)
12. Hu, Y., Panda, B.: Design and Analysis of Techniques for Detection of Malicious Activities in Database Systems. Journal of Network and Systems Management 13(3), 269–291 (2005)
13. Lee, W., Fan, W., Miller, M., Stolfo, S.J., Zadok, E.: Toward Cost-Sensitive Modeling for Intrusion Detection and Response. Journal of Computer Security 10(1-2), 5–22 (2002)
14. Debar, H., Dacier, M., Wespi, A.: Towards a Taxonomy of Intrusion-Detection Systems. Computer Networks 31(8), 805–822 (1999)
15. Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Webster, S., Wyschogrod, D., Cunningham, R., Zissman, M.: Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation. In: Proc. 2000 DARPA Information Survivability Conference and Exposition, vol. 2, pp. 12–26 (2000)
16. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to algorithms, 2nd edn., pp. 770–821. MIT Press, Cambridge (2001)
17. Grinstead, C., Snell, L.: Introduction to Probability, 2nd edn., pp. 325–360. American Mathematical Society, Providence (1997)

Secure Method Calls by Instrumenting Bytecode with Aspects

Xiaofeng Yang and Mohammad Zulkernine

School of Computing, Queen's University
Kingston, Ontario, Canada, K7L 3N6
{yang,mzulker}@cs.queensu.ca

Abstract. Today most mobile devices embed Java runtime environment for Java programs. Java applications running on mobile devices are mainly MIDP (Mobile Information Device Profile) applications. They can be downloaded from the Internet and installed directly on the device. Although the virtual machine performs type-safety checking or verifies bytecode with signed certificates from third-party, the program still has the possibility of containing risky code. Inappropriate use of sensitive method calls may cause loss of personal assets on mobile devices. Moreover, source code is not accessible for most installed applications, making it difficult to analyze the behavior at source-code level. To better protect the device from malicious code, we propose an approach of bytecode instrumentation with aspects at bytecode level. The instrumentation pinpoints the location of statements within methods, rather than at the interface of method calls. The aspects are woven around the statement for tracking. The weaving is performed at bytecode level without requiring source code of the program.

Keywords: Bytecode instrumentation, aspects, code security.

1 Introduction

Mobile technology is very popular nowadays. More and more applications are running on mobile devices such as cellphones and PDAs, etc. Mobile code can easily migrate from one site to another, and even to a personal cellphone. Java applications are popularly running on mobile devices that embed Java Runtime Environments. User's information may be at risk as they are not usually aware of the programs they are running on their devices, for example, the unauthorized SMS (Short Message Service) sending programs [1] can be executed without any authorization of a user. Such sensitive information may also include personal data, geo-locations, and passwords [18].

The MIDP (Mobile Information Device Profile) is the core profile for the development of applications on Java mobile devices. The MIDlet is the application developed with MIDP. Many cellphone manufacturers already have MIDP enabled devices available. The security vulnerabilities shown in [1] indicate that the incompatibilities between different versions of MIDP caused security issues

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 126–141, 2009.

by unprotected use of sensitive method calls. It is necessary to track such sensitive method calls to avoid abuse of APIs before the MIDP specification becomes perfect. Much research have been using static analysis techniques to assess the quality and the security of downloaded Java mobile applications [10,11,17]. They present approaches of validating downloaded MIDlets by verifying source code, method properties and signatures with certifications using Point-to analysis [10] or dependency analysis [11, 17]. Although these static analysis techniques are helpful, they cannot fully address the problems occurred during runtime. By static analysis, it is hard to tell which method call is at risk. For example, `Display.setCurrent()` is a common method used to update the screen on mobile devices. However, it was also used by hacker group to run malicious code on devices. The current screen is obscured with other items asking a user for permitting SMS. Then the user approves the permission for the request shown on the obscured screen rather than the real screen. Although this vulnerability is prevented in Sun RI (Reference Implementation) of MIDP by performing appropriate locking and unlocking access to the display, as there could be other sensitive method calls, we believe that it is necessary to control such method calls. Moreover, by static analysis, the method call `RecordStoreFile.deleteFile()` is allowed for use. However, this deletion operation can cause data loss during runtime with the inappropriate use of the developers as reported in [1]. Both examples show that such method calls should be controlled even though they are declared valid in the static analysis phase.

Aspect-oriented technology has been widely used to separate cross-cutting concerns (like logging mechanisms) from functionality modules. Currently, most implementations of aspect languages have been using the recompilation of the original source code and the instrumentation code to perform the instrumentation. However, in real world, most programs are delivered without source code. Therefore, we consider the instrumentation at bytecode level. Bytecode is an intermediate machine language that instructs virtual machines how to execute operational instructions. When the bytecode is loaded into JVM, the JVM verifier only validates the correctness of the bytecode's structure, data type, and symbolic references. BCEL (ByteCode Engineering Library) is used to manipulate and modify bytecode [20] at bytecode level, but it does not support aspect features for instrumentation and requires good expert knowledge to instrument bytecode at appropriate location. AspectJ is a popular AOP (Aspect-Oriented Programming) language [6] that implements aspects in Java. It provides the flexible aspect constructs and expressive pointcut patterns. It requires the source code of the program for re-compiling. Binder *et al.* [14, 15, 16] propose a series of approaches of bytecode re-engineering by instrumenting Java's standard class libraries. However, it is not realistic to modify the implementation of system libraries. We cannot rely on the platform's upgrade since it is difficult to recall and patch up all platforms. For mobile platforms, the bytecode instrumentation was not performed using aspects for security concerns at bytecode level.

In our approach, we modify the program with aspects at bytecode level without accessing source code and without knowing the context of sensitive method

calls in advance. We manipulate the delivered bytecode without recompiling the original programs, and we instrument the bytecode instructions during class loading. We use a bytecode library, Javassist [3], to manipulate bytecode instructions from class files. We also define the corresponding aspects in modules using `expression editor` provided by Javassist. While loading `classes` into the virtual machine, security concerns are instrumented into the place where insecure method calls might happen. Our approach does not require access to source code and does not change the original class. Moreover, rather than weaving aspects at the interface level, we weave directly around the statement of the method call for tracking. It does not change the implementation of the method call but it changes the behavior of invoking the method call. Our approach determines the context of a sensitive method call dynamically, while the others have to know the names of methods that are using the sensitive method call in advance.

The rest of the paper is organized as follows. Section 2 presents the Java ME (Java platform, Micro Edition) technology, security risks on Java ME platform, bytecode instrumentation, and aspect-oriented technology. Section 3 demonstrates our approach of bytecode instrumentation with aspects to address sensitive method call concerns. Section 4 presents the implementation of our approach and experiments. Section 5 compares our work with the existing related work. Section 6 concludes the paper.

2 Background

2.1 Security Risks on Java ME Platform

Java programming language was initially invented for using in embedded systems. It is becoming more and more popular for mobile embedded systems and mobile devices. Java ME (Java Platform, Micro Edition) is a platform for the development of Java applications on mobile devices. Today, most mobile phones have a Java runtime environment embedded in the device. The overview of the layers of Java ME architecture for resource-constrained devices is shown in Fig. 1.

CLDC HI (Connected Limited Device Configuration Hotspot Implementation) is Sun's new high-performance Java virtual machine for embedded devices compared to KVM (K Virtual Machine), which in the past was widely deployed on Java ME [5]. CLDC (Connection Limited Device Configuration) defines the configuration for "constrained" devices with low memory, limited network connectivity, and user-interface capabilities. MIDP (Mobile Information Device Profile), on top of the configuration, provides rich APIs for the development of mobile applications, such as file handling, networking, and persistent storage. The MIDlet is the application designed with MIDP profile. The VM (Virtual Machine) on mobile devices provides a security model that is different from conventional Java. It does not control access through `SecurityManager` to enforce security policies as standard Java does. Java class files are properly pre-verified in a pre-verification process of the MIDP application's life cycle. Developers or testers are not allowed to design customized class loader during run-time, and

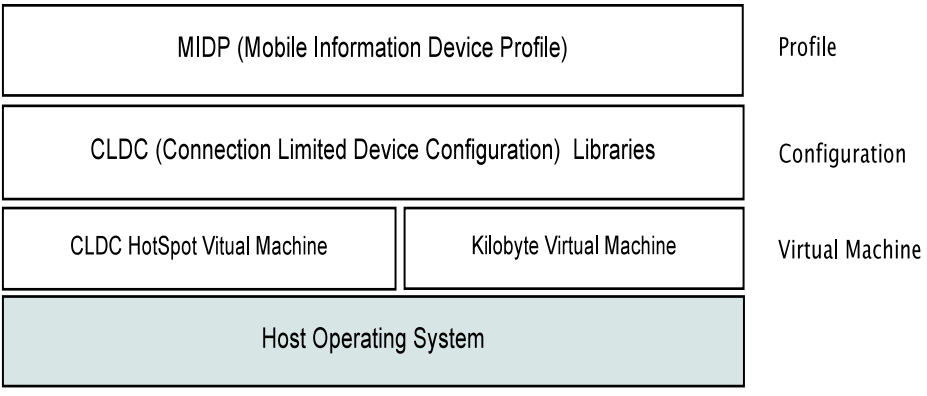


Fig. 1. Java ME Platform

it is also not allowed to download any new libraries containing native interfaces that are not supported in MIDP or CLDC.

From application-level considerations, the study of security vulnerabilities [1] reveals that the platform suffers from many problems caused by poorly written code, containing unauthorized method calls. As mobile devices hold very important information, it can be exploited easily as the devices roam. Applications running on Java ME platform are implemented with the core profile of MIDP. In MIDP 1.0 specification, the application is restricted into a sandbox model. The sharing between MIDlet suites is strictly prohibited. In MIDP 2.0, the access to sensitive resources such as method calls and sharing between MIDlet suites are allowed under granted permissions by the MIDlet. This can cause security problems by inappropriate usage of APIs. The program can access shared resources and facilities that contain sensitive method calls.

In MIDP 2.0, some low-level APIs should be protected from application developers. The application should not call such APIs via higher-level interfaces. For example, the class `RecordStoreFile` provides the direct interface of performing actions on data records in the storage system. In the transitional period of specification implementation, such APIs exist for the purpose of compatibility with old versions. However, with malicious designs, the data from other suites can be easily exploited by the methods provided in `RecordStoreFile` class [1]. Therefore, it is developers' responsibility to take care of all data protections or rely on the enhancement of MIDP security specifications.

Although manufacturers are implementing features complying with MIDP specifications, some conform to the 1.0 version, while some are already designing with the 2.0 version specification. Therefore, there still exists risks on devices because of the varieties of specifications. Meanwhile, MIDP specification has been enhanced with many security features [4]. Currently, the MIDP 2.0 is still the main profile for the implementations of applications on Java mobile devices. Considering these issues, we monitor related sensitive method calls in MIDP 2.0 applications.

2.2 Bytecode Instrumentation

Bytecode instrumentation is a technology used to manipulate and modify bytecode instructions with additional concerns integrated without impacting the original system behavior. It inserts user-defined classes or methods in the format of bytecode. Bytecode is a set of instructions that can be interpreted and executed by virtual machines. Once a Java file is compiled into the class file, the bytecode instructions in the class file can be transferred across the network to other platforms for executions. Java programs are compiled into a generic intermediate format called Java bytecode. A method in bytecode is a sequence of instructions. Each instruction consists of a one-byte operational code specifying the operation to be executed followed by one or more arguments. An instruction can be expressed in the following format [9]:

```
1 <index><opcode>[<operand1>[<operand2>...]][<comment>]
```

The `<index>` is the index of the opcode (operational code) of the instruction in the array that contains the bytes of Java virtual machine code for the method. It can be thought of a bytecode offset from the beginning of the method [9]. The following example is a simple constructor from a MIDlet application with a simple method function of `String.valueOf()`.

```
1 public MainMidlet() {
2     String number = String.valueOf("H");
3 }
```

The corresponding bytecode is shown as below. It consists of index and instruction code in each line. The opcode `aload_0 [this]` gets the reference of the class from the stack. The line 5 and line 7 are two instructions of method calls. The `invokespecial` invokes the method of an instance of class, `this`. The `invokestatic` invokes the static method of a class. Opcode `ldc` pushes the value of the String `'H'` onto the stack. The `astore_1` stores the value returned from the method call into the variable `number`.

```
1  // Method descriptor #10 ()V
2  // Stack: 1, Locals: 2
3  public MainMidlet();
4  0  aload_0 [this]
5  1  invokespecial javax.microedition.midlet.MIDlet() [12]
6  4  ldc <String "H"> [14]
7  6  invokestatic java.lang.String.valueOf(java.lang.Object) : java.lang.String [16]
8  9  astore_1 [number]
9  10 return
```

Bytecode instructions are checked by a bytecode verifier in JVM before loading to virtual machines. However, the bytecode verifier mainly checks the type safety in VM (Virtual Machine). It cannot predict the runtime behaviors of the instructions. Therefore, it is necessary to check additional properties of bytecode instructions and prevent them from imposing any harms on systems. Bytecode instrumentation uses the structural reflection to return the relevant information from the bytecode instructions. It is a good way to know what the instructions do at run-time.

Javassist is a tool that makes the manipulation of Java bytecode simple [3]. Javassist is a class library that deals with reading and writing Java bytecode. It takes great usage of Java reflection features to modify bytecode retrieved from class files. In Javassist, each Java class is represented by an abstract `CtClass` object. The `ClassPool` is the repository of all loaded classes. The typical process of the instrumentation by Javassist is as follows.

```
1 ClassPool pool = ClassPool.getDefault(); CtClass cc =
2 pool.get(targetClass);
3 //targetClass: String name of the class
4  ....
5 /* Modification Process */
6 cc.writeFile();
```

`ClassPool` represents the repository of all loaded classes. The `CtClass` represents the instance of the loaded class. All modifications on the class, such as inserting fields, changing method bodies, modifying constructors, are performed against the `CtClass` object. The method `writeFile()` commits the changes into the loaded `CtClass` object.

As the reflection features are not allowed on the Java ME Platform, the instrumentation is performed in Java standard virtual machine with MIDP libraries and verified before delivering to Java ME Platform.

2.3 Aspects

Aspect technology is used to separate crosscutting concerns from functionality modules. The most common cross-cutting concerns in a system is the logging mechanism. An *aspect* is a code snippet that scatters across multiple modules of the whole system. It aims to address crosscutting concerns by providing advices for systematic identification (*pointcut*), separation (*join point*), representation (*aspect*), and composition (*weaving*). A *join point* is a location where crosscutting concerns are attached to the original functionality. A *pointcut* is an expression language that specifies the patterns of locations where the code for cross-cutting concerns are required.

Crosscutting concerns are encapsulated into separate modules, known as *aspects*, so that security concerns can be developed separately from the existing programs [7]. Combining aspects and existing programs requires the *recompilation*

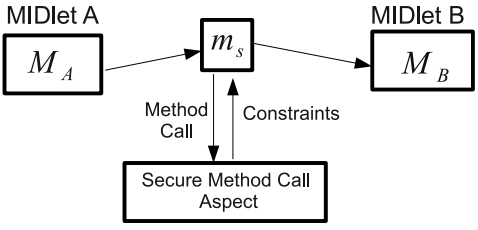


Fig. 2. Secure Method Call

process to form the final system. Using bytecode instrumentation, additional concerns and the original core program can be combined together without impacting the logical functionality of the original system.

3 Secure Method Call

We use aspect-oriented programming to track sensitive method calls during class loading. The join points are evaluated for the intercepted sensitive method calls. Currently, many existing approaches instrument security concerns by intercepting methods that use sensitive method calls in applications. However, a sensitive method call can be used everywhere by the application. The name or method signature can be assigned differently. Method context and class context should be known in advance to locate the secure method call. In our approach, the method context and class context are determined dynamically during runtime by the instrumented aspects. We insert tracking code directly around the method call without knowing which class or method will be using it in advance.

When a sensitive method call is requested, the appropriate checking is performed around the call. On both "caller" and "callee" sides, it should be verified whether the "caller" and the "callee" are in the granted permission of the request according to the policy by "callee" side. Secondly, the depth of method calls is evaluated to locate the origin of the method calls. In this way, we compare the chain of a method call with the specification to verify if the method call is in a safe state.

Let us consider a MIDlet suite containing two MIDlet applications, M_A and M_B. Assume that M_A retrieves the reference to one field of M_B, for example, the reference to the `record management system` in M_B. Then M_A can call method m_s, which is the sensitive method call as Fig. 2 shows.

The MIDlet M_A is performing actions defined by method m_s on MIDlet M_B. The M_A is the initiator of the method call. The sensitive method call invocation is intercepted by Javassist tool. The method call m_s is passed as a parameter to the secure method call aspect module. In the aspect module, the context of the method call is determined. The security state is evaluated by the constraints.

The security state of the method call is determined by the constraints returned from security state checking aspect. If the method call m_s passes the security

checks, the invocation is continued on the target M_B. Otherwise, the security warnings are prompted to user indicating failures of the security check. We encapsulate the "security state check" module into aspects. The aspects are instrumented around the sensitive method call. Bytecode instructions are instrumented into the context of the method call, rather than at the interface. We locate the context of the method call instead of changing method bodies, and we instrument concerns around the statement of the method call at the caller side.

The aspects are woven around the call and invoked when it is intercepted. At runtime, when the method call is invoked, the corresponding aspects are executed to check the security state of the call. The constraints returned from the aspect module decides whether to continue the method call. The security policies (permission check, domain check) are included in the aspects. The policies are returned as the constraints imposed on the execution of the method calls.

3.1 Bytecode Instrumentation Using Aspects

We design the corresponding aspects for concerns of sensitive method calls. The framework of the instrumentation process is shown in Fig. 3. The `ClassPool` is a repository representing all objects of the loaded classes. The `ClassLoader` is used to load classes from the pool into virtual machines. It takes the parameter of an instance of `ClassPool` to retrieve and manage properties of classes. The `Translator` is added to `ClassLoader` instructing how the classes from `ClassPool` should be loaded and when the instrumentation should be performed. It is implemented in the `onLoad()` method. The `onLoad()` method is invoked when the class is being read from `ClassPool`. The `ExprEditor` is the module containing aspect implementations. It implements how to modify the bytecode instructions when the `Translator` is going to instrument the class. In our framework, the target `Translator` performs instrumentation upon is the context (class and method) of the method call, rather than the body of the method call. As the custom classloader is not supported on Java ME platform, we use Java Standard Virtual Machine assisted with MIDP libraries to load the classes, manipulate the bytecodes, and instrument aspect codes. The modified bytecode are delivered to the mobile platform for final execution.

We use Javassist to encapsulate cross-cutting concerns. We design a subclass of `ExprEditor` class representing an `Aspect`. However, the description of an aspect library written with Javassist is not declarative but procedural whereas an aspect library in AspectJ is declaratively written with Java-like syntax [6].

A pointcut is defined by condition checks. In Javassist, it is specified by a function implemented in a class inherited from `ExprEditor` class: `edit(Expr e)`. The `edit()` method takes different parameter types as inputs including `FieldAccess`, `MethodCall`, and `ObjectConstruction`. For example, if we instrument on one field of an object, then the method `edit(FieldAcess f)` is defined to implement the condition checks on the field variables and specifies when the aspect is invoked. To design the pointcut with the combination of multiple types of fields, multiple `edit()` methods are defined in the class with different parameter types.

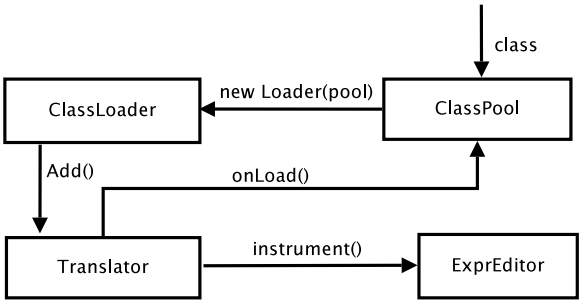


Fig. 3. Instrumentation Process

The join point is a process of determining where to instrument bytecode. Javassist uses Java reflection features to manage Java bytecode instructions. Unlike expressive patterns in AspectJ, the joinpoint determination has to be performed using procedural programming with reflection interfaces. After we intercept the method call, we determine the method and the class from which the method call is invoked. We further locate the position of the call statement within the method context and class context. Javassist instruments the statement when it is reached during loading.

The advice is the action performed at join points. We use Javassist to retrieve bytecode information and modify bytecode during loading them to the class loader. The advice is wrapped as a string of Java statements to replace the original statement. The original statement is the statement of executing the method call. The aspects are defined as separate method functions in `ExprEditor` class for modularity. All aspect codes are appended into a `StringBuffer` object, including the execution of the original method call.

3.2 Secure Method Call Aspect

We use the example shown in Fig. 2 to demonstrate the process of instrumentation in aspects at bytecode level. The method call is initiated from class M_A. If it invokes the `deleteRecord()` method call to access the object entity specified by `rs`, which is a `RecordStore` object in class M_B, the identity of the calling class M_A needs to be verified upon the call. In this case, the method call `deleteRecord()` needs to be intercepted. Class M_A is the class context containing the use of the method call. If the method call is used within a method in M_A, for example, `call()` method defined in class M_A, then `call()` is the method context of `deleteRecord()`. The context of the method call `deleteRecord()` is determined dynamically by aspects without knowing the name of context (class and method) in advance. This is different from other traditional approaches [7], in which the intrumentation actually happens in the body of the method call of `deleteRecord()`. Determing the context of a method call is to locate the position of statements of invoking the method call, that may be in the middle of the method context. The statement of the method call is then replaced with the instrumented code.

Pointcut. Sensitive method calls are intercepted statically by the Javassist tool during reading classes from `ClassPool`. The `Pointcut` specifies the interceptor patterns, indicating where sensitive method calls are happening. In our approach, we define the corresponding `edit(MethodCall m)` method to capture predefined sensitive method calls. When the method call `RecordStore.deleteRecord` is encountered, an `aspect` is instrumented into the context of the call. The `pointcut` is implemented by the method `edit()` in the class inherited from `ExprEditor`. The method function `edit()` contains the determination of point cut. It invokes the corresponding instrumentation containing join points and advice defined in `ExprEditor`. It takes different parameter types including `FieldAccess` and `MethodCall` to process different types of objects. If it targets on the type of method call, it then takes the `MethodCall` as the parameter to intercept method calls filtered by condition checks.

```
1  public void edit(MethodCall m) throws CannotCompileException{
2      if(matchMethod(m)){
3          try{ rmsInstrument(m);
4          }catch(Exception e){ e.printStackTrace();}
5      }
6  }
7
8  public boolean matchMethod(MethodCall m){
9      return (m.getClassName().equals(m_CName) && m.getMethodName().equals(
           m_MName));
10 }
```

The interception of the method call of `RecordStore.deleteRecord` is shown as below. The field member variable `mName` specifies the name of the method call. The method `matchMethod()` is a function to verify the signature of the method call. The variable `m_CName` refers to the class name defining the method call. In our example, it is `RecordStore`. The variable `m_MName` refers to the method name, which is `deleteRecord()`. The method `rmsInstrument()` is the aspect containing join points and advice to weave. It is defined in `ExprEditor`.

JoinPoint. The `JoinPoint` specifies the place where aspects are woven into. The process is to determine the context containing the statement of the intercepted method call. When the context is located, we instrument tracking aspects into it around the statement of the call. The method call is passed as a parameter to assess its context. If the intercepted method call is invoked from another method, then this calling method is the method context of the call, and the class that the calling method belongs to is the class context of the call. All the context information such as calling method, calling class, and calling MIDlet determine whether the caller is authorized for the call.

We first retrieve the method and the class containing this method call. Below is the function to retrieve the information of class context and the method context. The `CtClass` represents a class object, and the `CtBehavior` represents a block object that containts the use of `Expr` object.

```
public String getContextMethod(Expr expr){
    CtBehavior source = expr.where();
    return source.getName();
}

public String getContextClass(Expr expr){
    CtClass cc = expr.getEnclosingClass();
    return cc.getName();
}
```

After we get the class context and method context, we further locate the position of the statement of the method call. The advice is invoked as a method defined for the determined context, which is `CtMethod` shown in the example below:

```
final String className = getContextClass(m);
final String methodName = getContextMethod(m);
ClassPool cp = ClassPool.getDefault();
CtClass cc = cp.get(className);
CtMethod cm = cc.getDeclaredMethod(methodName);
cm.instrument(editor);
```

The `editor` is the "aspect" module instrumenting bytecode into the context of the method call `cm`. When reaching the method context `cm`, the `editor` further locates the position of the statement of method calls. The aspects defined in `editor` are instrumented when the statement is reached.

Advice. The `Advice` is the action performed as a response to the events of the sensitive method call. It is defined in instrumented code. In our example, it is encapsulated in the sub class of `ExprEditor`. After the class context and method context are determined, the next step is to construct the bytecode to weave into appropriate locations.

We design aspects around the statement of sensitive method calls appearing in their class and method context. We replace the method call statement by new bytecode containing logging concerns and the original statement. The new statements are encapsulated into `StringBuffer` object, including the executions of original statements represented by `$proceed()`.

```
    StringBuffer codes = new StringBuffer();
    codes.append("Log("before invoking the method call");");
    codes.append("$_ = $proceed();}");
    codes.append("Log("after invoking the method call");");
    m.replace(codes.toString());
```

Aspect Rule. To better refine the control flow of sensitive method calls, we adopt an aspect rule to control how the method call is granted access. The access rule is defined to confine the use of sensitive method calls that are risky to execute. The aspect rule is used to filter the class or the method to be instrumented.

```
1 <aop>
2     <pointcut name="rms" expr="execution(public * *.RecordStore->
          deleteRecord(int))"/>
3
4     <bind pointcut="rms">
5         <interceptor class="RmsEditor"/>
6     </bind>
7 </aop>
```

Each aspect is specified by an `aop` element. The `aop` element contains an element of `pointcut` and an element of `bind`. The `pointcut` specifies the pattern of the pointcut. It contains an attribute of `expr` that defines the expression of intercepted method call. In this example, it indicates an expression to intercept the method call of `RecordStore->deleteRecord(int)`. The method call can be initiated from any classes. The advice is defined in `bind` element. The `bind` element binds the `pointcut` and `interceptor`. The interceptor specifies the instrumentation class of `ExprEditor`. In this example, `RmsEditor` is a subclass of `ExprEditor` containing aspects implemented by its method `edit()`. When the method call specified by `pointcut` is captured, the advice specified by `interceptor` element, `RmsEditor` is invoked to instrument the method call.

4 Implementation

The instrumentation framework is written in Java. The UML diagram of the implementation is shown in Fig. 4. We use one MIDlet application (`RmsApp`) of RMS (Record Management Store) as a target application. The target application contains the usage of the sensitive method call of `deleteRecord()`. We instrument the designed aspects into the application around the sensitive method call for tracking purpose. The modified bytecode instructions run on an emulator device. Whenever a sensitive method call is invoked, the event is logged with the source of initiator. This approach provides a way to track and locate the source of malicious code on the device.

The `RmsTranslator` is constructed to control how class files are interpreted. The translator class is attached with the `RmsEditor` which contains the instrumentations in aspects. The `RmsSecureController` is the entrance point of the instrumentation process.

Class `RmsSecureController` contains three members. ClassLoader is a user-defined class loader which is used to load classes. `ClassPool` represents the repository of all objects read from class files. `ClassLoader` reads the classes from `ClassPool`, as we have shown the process in Fig. 3. `RmsTranslator` provides instructions to `ClassLoader` about how to interpret and load classes at runtime.

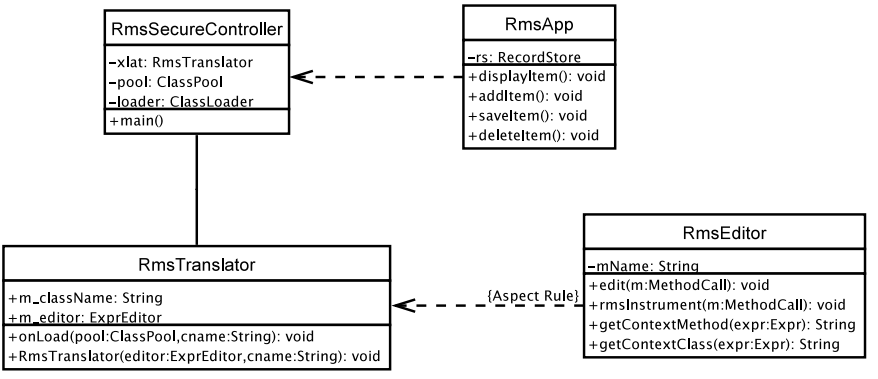


Fig. 4. UML of Instrumentation Class

Class `RmsTranslator` defines the `onLoad()` method. The method implements the function of loading the required method calls. It has two parameters, `ClassPool` object, and the string indicating patterns to match the required method call. The constructor of `RmsTranslator` has two member attributes: the class name variable indicating the target name to intercept and the `ExprEditor` specifying expressions of patterns for identifying join points.

Class `RmsEditor` encapsulates aspects into method implementations. The `edit()` method is designed to intercept method calls specified by its parameters. When the desired method call is intercepted, the aspects are invoked by `rmsInstrument` around the positions of statements. The positions of statements are calculated by `getContextMethod()` and `getContextClass()` to retrieve the context information including the class and method contexts.

We verify whether the instrumentation is successful in two phases. In the first phase, we check the bytecode instructions statically. In instrumentation process, modified bytecode are written into temporary files. Using `javap` program and the Eclipse IDE to disassemble the class files, the bytecode of `Log()` methods are instrumented successfully around the `deleteRecord()` method call. In the second phase, we conduct the run time checking. The modified bytecode is delivered to the emulator toolkit at run time. When a user invokes the `deleteRecord` operation, the logs are recorded specifying where the operation is initiated from and where it is reached. The bytecode is instrumented for tracking the source of the call.

Our instrumentation is conducted in Java Standard Environment and tested using Sun's Java Wireless Toolkit 2.5.2 for CLDC on Debian GNU/Linux operating system. During the instrumentation, we set the `bootclasspath` to Java ME library version to make it deliverable to mobile devices. As the instrumentation is performed in Java Standard Environment assisted with MIDP libraries, we use the `preverify` command tool provided by Java Wireless Toolkit to statically verify the validity of modified classes. Since the instrumented aspect code are only limited to APIs allowed only on Java ME platform, the classes pass through a pre-verification process.

5 Related Work

Static analysis technique is commonly used to assess the quality and the security of downloaded programs. The MATOS (Midlet Analysis TOol Suite) [10] automatically validates MIDlets with Point-to analysis. It analyzes properties of JAD (Java Application Descriptor) conformity, signature certificates, allowed usage of classes and methods, and acceptable ranges of argument values. Bian *et al.* [11] also perform bytecode analysis and combine the technique of dependency analysis with the information flow analysis to predict secure information flow from bytecode's composition. Avvenuti *et al.* [17] propose a security mechanism with the static analysis to indicate the state of secure information flow. All of these static analysis techniques are helpful in static checking phase, but they cannot address all the issues that may be found at runtime. In [19], a formal specification of dynamic semantics of Java bytecode is presented in the format of operational semantics for the Java Virtual Machine, giving each instruction a rule describing its effect based on the machine state. These approaches provide a static view of assessment of bytecode security. Our approach protects the use of sensitive method calls and tracks their behavior during runtime. It is more practical because most mobile applications are deployed in the format of bytecode.

Bytecode instrumentation has been used in the area of debugging and profiling. Binder *et al.* [14, 15, 16] propose a series of approaches of bytecode re-engineering by instrumenting Java's standard class libraries. The bytecode instrumentation has also been used to adapt the legacy software to distributed execution on multiple JVM [13] migrated from the version running on single JVM. However, it is not realistic to modify the implementation of system libraries, although source code is available for some systems. We cannot rely on platform's upgrade since it is difficult to recall and patch up all platforms currently in use. Our approach neither modifies the platform of applications, nor changes the original programs. We instrument the bytecode instructions while loading applications into virtual machine. The bytecode is modified during loading. After the execution is completed, the program remains as it was when downloaded.

Many tools assist in flexible bytecode instrumentation. BCEL [20] (ByteCode Engineering Library) provides libraries to manipulate low-level bytecode. It directly operates on the instructional operands. It requires proficient skills on bytecode. Win *et al.* [7] illustrate the effectiveness of instrumenting separate application security concerns using AspectJ. However, they either require the source code for instrumentation, or instrument bytecode instructions in a procedural way. Our approach uses Javassist to modularize concerns in aspects. Javassist operates on Java bytecode with flexible expressions. It combines the benefits from both BCEL [20] and AspectJ [7]. Hence, we can integrate more concerns flexibly. It does not require too much knowledge of low-level bytecode instructions or the access to source code.

The applet filter was proposed [12] by substituting the references of potential classes and methods. The authors provide a solution to change the references of restricted classes and methods stored in the `CONSTANT_POOL`, where the references

of all loaded method calls are stored. However, if there are more unsafe classes or methods, more subclass and method references should be designed. They also change the standard class libraries provided by JVM environment. It is not realistic to modify the platform. In our approach, we do not modify the platform and only focus on the application-level. We do not modify the implementation of a program, but weave the additional security concerns without changing the functionality of the original program.

6 Conclusion

Without the source code of applications, it is difficult to manipulate bytecode in a flexible way. We present an approach of bytecode instrumentation using aspects. The process does not require access to source code. We design corresponding aspects to track secure method calls by weaving it into executable applications without having too much knowledge about bytecode. The bytecode instrumentation is performed during class loading. Therefore, we do not need to change the original classes. The bytecode instructions are inserted before and after the intercepted method call successfully to track the call event. Determining the contexts of the sensitive method calls dynamically is another benefit of our approach, while existing approaches require the names of methods that use the sensitive method calls in advance.

In future, we will try to make aspects more flexible to instrument. Javassist is a procedure-based facility for bytecode instrumentation. The modularized aspects are still not quite flexible and obscure to understand. Therefore, the AOP features provided by Javassist is kind of limited. However, the principle of AOP for bytecode instrumentation is the same. Currently, we address only the security issues related to sensitive method calls. We plan to use more security rules to integrate more flexible security concerns.

Acknowledgments

This work is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Debbabi, M., Saleh, M., Talhi, C., Zhioua, S.: Vulnerability Analysis of J2ME CLDC Security. US DoD Information Assurance Newsletter 9(2), 18–23 (2006)
2. Debbabi, M., Saleh, M., Zhioua, S.: Java for Mobile Devices: A Security Study. In: Proceedings of the Annual Computer Security Applications Conference, ACSAC 2005, Tucson, Arizona, USA. IEEE Press, Los Alamitos (2005)
3. Javassist, `http://www.csg.is.titech.ac.jp/~chiba/javassist/`
4. JSR 271: Mobile Information Device Profile 3, `http://jcp.org/en/jsr/detail?id=271`

5. Sun Java ME CLDC HotSpot Implementation White Paper, http://java.sun.com/products/cldc/wp/CLDC_HI_WhitePaper.pdf
6. AspectJ Programming Guide, http://www.eclipse.org/aspectj/doc/released/proggui-de/index.html
7. Georg, G., Ray, I., France, R.: Using Aspects to Design a Secure System. In: 8th Int'l Conf. on Engineering of Complex Computer Systems, pp. 117–128 (2002)
8. Using Javassist for bytecode search and replace transformations, http://www.ibm.com/developerworks/java/library/j-dyn0302.html
9. Lindholm, T., Yellin, F.: The Java Virtual Machine Specification, 2nd edn. Addison Wesley, Reading (1999)
10. Crégut, P., Alvarado, C.: Improving the security of downloadable Java applications with static analysis. In: BYTECODE. ENTCS, vol. 141. Elsevier, Amsterdam (2005)
11. Bian, G., Nakayama, K., Kobayashi, Y., Maekawa, M.: Java Mobile Code Security by Bytecode Analysis. ECTI Transactions on Computer and Information Technology 1(1), 30–39 (2005)
12. Chander, A., Mitchell, J.C., Shin, I.: Mobile code security by Java bytecode instrumentation. In: DARPA Information Survivability Conference & Exposition (DISCEX II) (June 2001)
13. Tatsubori, M., Sasaki, T., Chiba, S., Itano, K.: A bytecode translator for distributed execution of legacy java software. In: Knudsen, J.L. (ed.) ECOOP 2001. LNCS, vol. 2072, pp. 236–255. Springer, Heidelberg (2001)
14. Binder, W., Roth, V.: Security Risks in Java-based Mobile Code System. Scalable Computing: Practice and Experience 7(4), 1–11 (2006); SWPS
15. Binder, W., Hulaas, J., Moret, P.: Advanced Java Bytecode Instrumentation. In: 5th International Conference on Principles and Practices of Programming in Java, Lisbon, Portugal, pp. 135–144 (2007)
16. Binder, W., Hulaas, J., Moret, P.: Reengineering Standard Java Runtime Systems through Dynamic Bytecode Instrumentation. In: Seventh IEEE International Working Conference, September 30, pp. 91–100 (2007)
17. Avvenuti, M., Bernardeschi, C., De Francesco, N.: Java bytecode verification for secure information flow. ACM SIGPLAN Notices 38(12) (December 2003)
18. Resource and Information Flow Security Requirements for MOBIUS (Mobility, Ubiquity and Security) (2006), http://mobius.inria.fr/twiki/pub/DeliverablesList/We-bHome/Deliv1-1.pdf
19. Bertelsen, P.: Dynamic semantics of Java bytecode. In: Workshop on Principles on Abstract Machines (September 1998)
20. The Byte Code Engineering Library (BCEL) manual, http://jakarta.apache.org/bcel/manual.html

Distributed Privilege Enforcement in PACS

Christoph Sturm[1], Ela Hunt[2], and Marc H. Scholl[3]

[1] Department of Informatics, University of Zurich
sturm@ifi.uzh.ch
[2] Computer and Information Sciences, Strathclyde University
ela.hunt@cis.strath.ac.uk
[3] Department of Computer & Information Science, University of Konstanz
marc.scholl@uni-konstanz.de

Abstract. We present a new access control mechanism for P2P networks with distributed enforcement, called P2P Access Control System (PACS). PACS enforces powerful access control models like RBAC with administrative delegation inside a P2P network in a pure P2P manner, which is not possible in any of the currently used P2P access control mechanisms. PACS uses client-side enforcement to support the replication of confidential data. To avoid a single point of failure at the time of privilege enforcement, we use threshold cryptography to distribute the enforcement among the participants. Our analysis of the expected number of messages and the computational effort needed in PACS shows that its increased flexibility comes with an acceptable additional overhead.

1 Introduction

Peer-to-Peer (P2P) networks are used in many application areas. Especially in the data distribution (file sharing, IPTV) and data integration area including Peer Data Management Systems (PDMS) [1], their use introduced new challenges for access control. Because traditional access control mechanisms are based on a central authority, which is missing in a P2P environment, new approaches are needed.

In previous work [2,3] we proposed an access control system for PDMSs without data replication. Here, we extend the application range of our P2P Access Control System (PACS) by supporting replication. Replication is a core feature in P2P applications. It enables fault tolerance in the face of disappearing peers and ensures load balancing within the network.

Replication in a P2P environment has several implications for the access control mechanism. First, as there is only limited trust between the participants, replicas cannot be distributed in plain text. The data is encrypted and stored on the replica servers without giving the replica server the possibility to decrypt it. This method is widely used in database-as-a-service approaches e.g. [4,5]. Second, the enforcement of privileges has to be distributed, as otherwise the data owner remains the single point of failure and a potential bottleneck. Again, because of limited trust between the participants, the enforcement cannot be transferred to a single peer.

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 142–158, 2009.

The contribution of this paper is to develop a distributed privilege enforcement mechanism for P2P networks, by extending PACS to support data replication. This implies the distribution of privilege management and privilege enforcement. The distribution of privilege management is already a key feature of PACS. Therefore, the main contribution here is to distribute privilege enforcement. To provide this, we use client-side enforcement, where each data object is encrypted with its own private key. The distribution and generation of these keys is done collaboratively by a group of peers (elected by the data owner) using threshold cryptography [6]. In this way, powerful access control models like Role Based Access Control (RBAC) [7] and administrative distribution are supported.

The paper is organized as follows. In Section 2 current access control approaches for P2P networks are presented. Section 3 introduces the access control component of PACS, together with the requirements for the data storage and the threat model. Section 4 describes the details of privilege enforcement in PACS. Thereafter, cost and performance estimations of the proposed enforcement mechanism are shown. The paper concludes with a discussion and future work.

2 Related Work

Research projects that address access control in P2P networks can be classified according to their distributed privilege enforcement method, as presented in the following.

The first option is to burden the data owner with privilege enforcement. This matches the classical server-side enforcement as the data owner is always the data provider. Enforcement is distributed in the network since the individual data owners, rather than a central authority, enforce the privileges on their data. When the data is replicated the enforcement of the privileges on that data must be done by the replication server. In a P2P network this is rarely practical, as trust between the participants is limited. Projects that use this enforcement technique are P-Hera [8], Berket et al. [9] and our own approach PACS [2,3].

The second option is to enforce the privileges on the client. In this case, the data owner encrypts the data before it is stored on the replica servers. Only clients that possess the corresponding key can decrypt the data delivered by replica servers. The decryption keys are distributed by the data owner according to the privileges held by the clients (e.g. [5]). P2P approaches based on client-side enforcement were developed for P2P file systems. Sirius provides read and write access control [10]. Pacisso [11,12] additionally supports owner changes and the possibility to define object groups. Both approaches operate at file granularity and do not support advanced access control models such as RBAC and administrative distribution. The access control mechanism proposed by Miklau and Suciu [4], designed for simple P2P file sharing, is based on encryption and key distribution. The privileges are defined using a logical model so that the granularity is not restricted to files. Again, no administrative distribution or definition of groups and roles is supported. In client-side enforcement

efficient key management is a problem, as a revoke triggers data re-encryption. Additionally, privileges are bound to decryption keys, which limits the access control models that can be supported. The advantage of this option is that replication of confidential data is supported even in unreliable environments.

Sandhu et al. [13] use trusted computing technology for access control in P2P networks to avoid the shortcomings of cryptographic enforcement. Their solution is also the problem, as a trusted reference monitor on the client is mandatory.

The third option is to transfer enforcement to a group of peers, called delegates in the following. The data owner elects delegates which enforce the privileges for a data object on its behalf. The delegates execute a t of n (where $t < n$) voting scheme to decide collaboratively whether a particular request should be permitted or not. The client can only access the data if t of n delegates approve the request. This mechanism strengthens the robustness of the enforcement mechanism and reduces the trust required in single delegates. Threshold decryption techniques [6] do not disclose the decryption key to the delegates or the clients. The delegates do the data decryption collaboratively and can enforce arbitrary privileges. The disadvantage is that the data has to be decrypted and delivered to the client by at least t delegates, which causes high communication costs. Pacisso [12] uses threshold signatures to enforce write permissions and Saxena et al. [14] show the use of this technique especially for mobile ad hoc networks. In addition, several research systems use threshold cryptography for access control enforcement in distributed databases (e.g. [15]), but none of them is applicable to P2P networks.

None of the presented approaches fulfills our requirement to support distributed enforcement for a powerful access control model and administrative delegation.

3 The Access Control Component

We first introduce PACS and then describe the distributed enforcement mechanism which is the main contribution of this paper. Our description abstracts from concrete access control models and data storages. In the following a user corresponds to a peer.

3.1 P2P Access Control System

PACS [2,3] enables peers in a P2P network to establish global, decentralized access control. The access control mechanism is built bottom up, as single peers grant each other privileges on their private data. Through administrative delegation, privileges for non local data can also be granted. The privileges are stored in a distributed reliable directory, based on Castro et al. [16]. This guarantees that data stored in the privilege store is always available. PACS supports RBAC with administrative delegation and does not rely on any central authority or component. To make this possible, each participant needs a certificate

signed by a certification authority. The established public key infrastructure (PKI) enables secure authentication and ensures secure communication between the participants.

3.2 PACS Distributed Enforcement

To enable distributed enforcement in PACS, we propose a combination of delegate enforcement and client-side enforcement, as introduced in Section 2. The delegates do not decrypt the data object, but the data object decryption key. In that way we maintain the flexibility to enforce a variety of privileges and enable administrative distribution. The drawback is that clients get the decryption key and, therefore, the key generation and data re-encryption problem remains. In our solution, key distribution and key generation is done by the delegates collaboratively while data re-encryption is handled by the client that revokes a concrete privilege. We lose some of the flexibility of the delegate option, as we cannot use contextual information for authorization. Especially, it is impossible to support the evaluation of attributes during XACML [17] policy evaluation. These attributes can change anytime and the privilege enforcement component cannot keep track of these changes, as they are outside the scope of the access control component. Recall that every attribute change may result in privilege revocation, which must trigger new key generation and data re-encryption. In addition, if an application area relies on attribute evaluation, one can think of a wrapper that maps this attribute value to a permission inside PACS. In this way, PACS keeps track of the attribute changes and so key generation and data re-encryption can be triggered.

3.3 Data Storage

To make our approach more general, we separate privilege management and storage from data storage. Even though it is possible that both are the same, it is very likely that storing privileges requires higher security and is therefore more expensive. We make only basic assumptions about data storage. The storage has to provide a simple interface with two operations *put* and *get*. *put(ID, Data Object)* stores the data object under the given ID and *get(ID)* retrieves the data with the specified ID.

Internally, data storage can be based on a distributed hash table (DHT) [18] or any other storage structure. What is important is that each data object stored in the data storage is identified by a unique ID. As we operate in a P2P network, the network-wide data object ID consists of the unique peer ID of the data owner concatenated with the unique local name of the data object. The ID of the data object DO of peer $P1$ is defined as $ID_{DO_{P1}} = ID_{P1} + ID_{DO}$. To ensure basic security in the data store, data objects to be protected by PACS are inserted with a self-verifying identifier, ID^{SV}. An ID^{SV} for a data object is generated by creating a hash value over the concatenation of the object content and the object header $head_{DO_{P1}}$ as $ID^{SV}_{DO_{P1}} \equiv h(head_{DO_{P1}} + DO)$, where h is a secure hash function, such as SHA-1. The peers in the data store ensure that only data

objects with a correct ID^{SV} are stored. Clients that receive a requested data object can immediately verify whether the data object was changed without authorization.

Data objects protected by PACS have the following structure:

$$[head\{ID_{DO_{P1}}, ID_{DGIO}, V_{OKey_{DO_{P1}}}\}, DO]$$

Beside data content (which is normally encrypted), they carry a header that contains the $ID_{DO_{P1}}$, the ID of the Delegation Group Info Object (DGIO, explained in Section 4.3) and the version number of the object encryption key. This information is needed to check valid object deletions. As the self-verifying identifiers change with every data object modification, a special delete option is needed for outdated data objects. A deletion of DO is invoked by a $put(ID^{SV}_{DO_{P1}}, DGIO)$ call. Note that the verification of this request fails, as $ID^{SV}_{DO_{P1}} \not\equiv h(DGIO)$. But if the object is a new DGIO with the same ID_{DGIO}, including a new version number that is larger than the one stored in the current object, the request is accepted as a valid deletion. Before that, the correctness of the delivered DGIO object has to be checked by requesting the DGIO object from the privilege store.

3.4 Malicious Peers

As PACS is based on a P2P network that potentially contains malicious peers, we distinguish between good and malicious peers: a good peer always follows the specified procedures, otherwise it is a malicious peer. A malicious peer can act arbitrarily wrongly and can collaborate with other malicious peers. Data owners are assumed to work correctly as far as their own data objects are concerned. Authorized peers that revoke a privilege are assumed to execute the procedure properly. Only then the revocation is effective, which is in their own interest.

As PACS is based on secure routing [16], it can handle up to 30% malicious peers in the network. We assume that the fraction of malicious peers within a delegation group equals the fraction of malicious peers in the network. It is the responsibility of the data owner to choose the delegates and the size of the delegate group n accordingly. In addition, PACS requires that the threshold $t \geq \frac{n}{2}$ as a security buffer.

The goal of an attacking malicious peer is to gain unauthorized access to stored objects and/or to prevent other authorized peers from accessing stored objects. Attackers are assumed to have only limited computational resources and hence cannot break the underlying cryptographic schemes. This means that an attacker cannot generate valid signatures and encrypt/decrypt data objects without having the right keys. An attacker can eavesdrop the network traffic as long as it is not encrypted, but cannot block communication between any two peers. We ensure that the communication between two peers is secure, by signing and encrypting messages. Challenge-response rounds are used to prevent replay attacks. Furthermore, we do not control the information flow after the data is delivered to the client. The data responsibility of PACS ends there.

4 Distributed Privilege Enforcement in PACS

Distributed privilege enforcement is based on a combination of symmetric, asymmetric and threshold cryptography [19], as presented below.

4.1 Object Encryption

Privilege granularity in PACS is on the object level. This is not a restriction, as the object can be defined as a tuple in a database table or a music file. Every data object o is symmetrically AES [20] encrypted by an individual object key $OKey_{o_p}$ (128 bit length), where p is the data owner. Each $OKey_{o_p}$ has a version number $V_{OKey_{O_P}}$. The data owner p provides the asynchronous private master key $MKey_p^{-1}$ and the corresponding public key $MKey_p$. The $OKey_{o_p}$ is generated by signing the concatenation of the data ID_{o_p} and $V_{OKey_{o_p}}$.

$$OKey_{o_p} = sig^{MKey_p^{-1}}_{(ID_{o_p}+V_{OKey_{o_p}})} = (h(ID_{o_p} + V_{OKey_{o_p}}))^{MKey_p^{-1}}$$

PACS uses RSA [21] as its signature algorithm. Enforcement is done by a delegation group. The data owner distributes the private master key $MKey_P^{-1}$ using Shamir's (t, n) secret sharing technique [22] [1]. Using this, we assign a share s_i to each delegate d_i. The secret is described by a polynomial $f(x)$ of degree of at most $t-1$, where $f(0) = MKey_P^{-1}$. The polynomial can be reconstructed by every combination of t participants, using the Lagrange interpolation formula. All coordinates x_i are made public, whereas the generated shares are private. Share $s_i = f(x_i), x_i \in \mathcal{R}$ is delivered to delegate d_i encrypted with its public key, so only d_i can read it. Based on Desmedt [19], a message m is then partially signed by delegate d_i which computes

$$sig_m^{s_i} = (h(m))^{s_i} (mod\ n')$$

where n' is the RSA modulus of the public key $MKey_p$. A full signature of m, $sig_m^{MKey_p^{-1}}$, can be calculated out of t partial signatures by first calculating the coefficients of the polynomial as

$$k_i = \prod_{i=1;j\neq i}^{t} \frac{x_j}{x_j - x_i}, \forall i \in [1, \ldots, t]$$

and then

$$sig_m^{MKey_p^{-1}} = \prod_{i=1}^{t} (sig_m^{s_i})^{k_i} (mod\ n')$$

[1] For RSA, some modifications are needed. See [23] for details.

Share Verification. Until now the generation of $sig_m^{MKey_p^{-1}}$ from the partial signatures fails if one of the t partial signatures is malicious. As we cannot verify the correctness of each partial signature on its own, it is very cumbersome to identify the malicious one and replace it with another correct partial signature from another delegate. To enable the verification of partial signatures, we use the robust threshold RSA method (see Gennaro et al. [24]). During share generation the data owner generates a random public sample message w and its corresponding partial signature $sig_w^{s_i}$ for all shares s_i. These partial signatures as well as the complete signature of the sample message $sig_w^{MKey_p^{-1}}$ are made public. The verification procedure is then as follows

1. The requester has received the partial signature of the message m, $sig_m^{s_i}$, from delegate d_i
2. The requester chooses two random numbers i, j and computes the challenge $Q = m^i \times w^j$ and sends this challenge to the delegate d_i
3. The delegate signs the challenge with its partial key $sig_Q^{s_i}$ and returns it to the requester
4. The requester now checks if $sig_Q^{s_i} = sig_m^i \times sig_w^j$. If this is true, the requester accepts $sig_m^{s_i}$ as partial signature of the delegate d_i

Share-Share Generation and Verification. A share s_i is only known to the delegate d_i and the data owner. When a delegate leaves the network unexpectedly and the data owner is not available, there is no possibility to restore the lost share. To cope with this, the so called share-shares are generated during share generation. For each share s_i a Shamir's (t, n) secret sharing schema is created. The procedure is as follows [12]

1. For each delegate d_i, the data owner takes its share s_i and applies the (t, n) secret sharing schema. The polynomial $f_i(x)$ is of degree $(t-1)$ and $f_i(0) = s_i$
2. The owner evaluates $f_i(x), x_i \in \mathcal{R}$ for n points $[x_1, \ldots, x_n]$ and gets as result n share-shares $[s_{i,1}, \ldots, s_{i,n}]$ with $f_i(x_j) = s_{i,j}$
3. The owner now sends to each delegate d_i the share-shares $[s_{1,i}, \ldots, s_{n,i}]$. So d_i gets as share-shares the points with the i^{th} x value from every polynomial. For each share-share, hash values are generated and publicly stored. They are used for share-share verification during share reconstruction

When a delegate d_j is no longer available and the share should be reconstructed, every remaining delegate d_i takes its share-share $s_{i,j}$ and sends it securely to the new delegate. After the new delegate has received t valid share-shares (the share-shares can be verified using the generated share-share hash values), it can reconstruct s_i using the Lagrange interpolation formula. Thereafter, the old delegates send the new delegate their share-share $s_{j,i}$. After receiving all share-shares, the new delegate has also recovered the share-shares of d_j.

4.2 Making a Data Request in PACS

The data request procedure is shown in Figure 1. We illustrate the case of the first request for a particular object made by a client. Note that we make a distinction

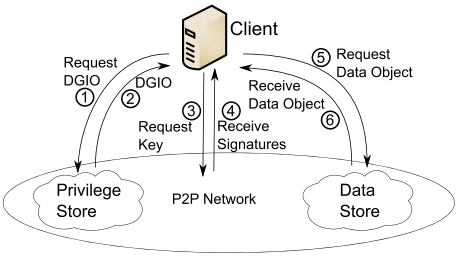


Fig. 1. A Data Request in PACS

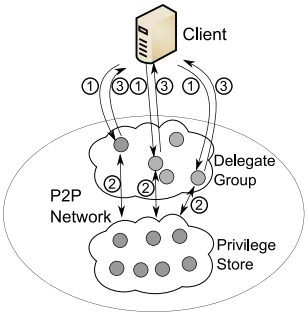


Fig. 2. Delegates Process an Access Request

between the privilege store and the data store. This is only a logical difference, as the two stores may run different storage protocols and different properties on the stored data can be guaranteed. So the same peers may participate in both logical networks. The client starts by sending a request for the Delegation Group Info Object (DGIO) for a particular object (Figure 1, step 1). The ID of the DGIO is derived from the object ID. The DGIO stores the information about the delegation group responsible for the object (for details see Section 4.3). Having received the DGIO, the client now knows which delegates to contact to get the decryption key. The decryption key in PACS corresponds to a threshold signature [6] over the object ID together with a version counter (cf. Section 4.1). The client has to ask a sufficient number of delegates to return a partial signature (steps 3 and 4). This is shown in detail in Figure 2. The client p sends the request to several delegates (step 1). Each delegate individually verifies the privileges held by p by requesting those from the privilege store (step 2). A good delegate returns the partial signature only if this verification succeeds (step 3). Privilege verification is executed by the delegate in the same way as a server does it in the sever side enforcement. After the client receives enough partial signatures, it constructs the complete signature that corresponds to the $OKey$ (see Section 4.1). Having the key, the client decrypts ID^{SV} from the DGIO and requests the encrypted data object from the data store. After receiving the object, the client decrypts the data and the request is satisfied. If the key is already known to the requesting client (caching), steps 3 and 4 in Figure 1 are skipped.

4.3 Delegate Management

Delegate management is part of privilege management. All information about the delegates is stored in a Delegation Group Info Object (DGIO). Because the information is as critical as the privileges stored in PACS, it is also stored in the privilege store. To achieve maximum flexibility, we allow a peer to have multiple delegation groups (e.g. one can think of a DGIO for every object). The peer just has to compute a unique master key pair $[MKey_p^{-1}, MKey_p]$ for every delegation

group. To keep the description simple, we assume for the rest of the paper that every peer has exactly one delegation group and therefore one delegation object. There exists an agreed naming schema for the DGIO so that every interested peer can compute the ID_{DGIO} of the right DGIO, using the data object ID.

Delegation Group Info Object. The DGIO object consists of a header part, an object part and a delegate part. The header is defined as:

$$[ID_{DGIO}, V_{DGIO}, sig_{DGIO}^{MKey_p^{-1}}, MKey_p, ID_p, PKey, w, sig_w^{MKey_p^{-1}}]$$

where $sig_{DGIO}^{MKey^{-1}}$ is the signature of the whole DGIO (excluding the signature itself), which prevents unauthorized changes. $PKey$ is the public key of peer p and w is the sample message needed for partial signature verification. As every delegation group has its own master key, the public master key $MKey_p$ is also included. V_{DGIO} is the DGIO version number required to detect stale DGIO copies.

The object part contains the object entries the delegation group is responsible for. A single object entry is defined as $[ID_{o_p}, enc_{ID_{o_p}^{SV}}^{OKey_{o_p}}, V_{OKey_{o_p}}]$. $ID_{o_p}^{SV}$ is the actual self-verifying identifier needed to retrieve o_p from the data store. It is stored in the DGIO to ensure that only authorized peers can create a new version of the object. $ID_{o_p}^{SV}$ is encrypted with $OKey_{o_p}$ to ensure that only authorized peers can see it. ID_{o_p} is stored for the $OKey_{o_p}$ creation together with a version number of the object key $V_{OKey_{o_p}}$.

The delegate part contains delegate entries. There is one delegate entry for each group member. A delegate entry is defined as:

$$[ID_{d_i}, DKey, x_i, sig_w^{s_i}, \{x_{i,1}, \ldots, x_{i,n}\}, \{h(f_i(x_{i,1})), \ldots, h(f_i(x_{i,n}))\}, active]$$

An entry includes the ID_{d_i} of the delegate (so that one can contact it directly), its public key $DKey$ and which x_i coordinate is assigned to it as share s_i. The share-share coordinates $x_{i,j}$ and the hash values of the share-shares are also stored here. Lastly, the status of the delegate (active/deleted) is noted.

Handling of Delegate Changes. Delegates are peers in a P2P network. They may disappear without informing the other delegates in their group. If a remaining delegate d_i realizes the absence of a delegate d_j in its group, it initializes a delegate recovery procedure. To achieve this, d_i initiates a quorum (again, t of n) decision by contacting all remaining delegates to get their agreement that

- delegate d_j is no longer available and should be marked deleted in the DGIO
- d_{n+1} should be the new delegate that substitutes d_j

d_j is marked deleted to exclude this peer as a future delegate substitute. Having received a positive quorum means that d_i has received t valid partial signatures for the new modified DGIO. Therefore, d_i can reconstruct the whole signature $sig_{DGIO}^{MKey_p^{-1}}$, as shown in Section 4.1. $sig_{DGIO}^{MKey_p^{-1}}$ is stored in the DGIO and the

signed new DGIO is transmitted to the delegate group members. Before it is stored by d_i in the privilege store, the signature recovery process (Section 4.1) is started, to recover share s_j for the new delegate d_{n+1}. As the number of delegates needed to recover a lost share is t, we need at least $t+1$ good delegates to recover a share of a lost good delegate.

A last note on security: In our solution the shares and the share-shares do not change. This means that the share s_j owned by the disappearing d_j is still valid. If d_j joins the network again, it needs only $t-1$ partial signatures $sig_m^{s_i}$, where $i \neq j$ to get a valid signature $sig_m^{MKey_p^{-1}}$. Consequently, frequent delegation group changes weaken the security of the threshold decryption and the secure sharing schema. Proactive secret sharing [25] avoids this by changing the shares and share-shares frequently. For our purpose, it is sufficient when the data owner reinitializes the secret sharing by generating new shares and share-shares, depending on its personal security requirements. Generating new shares changes only the DGIO and the delegates. There is no need to re-encrypt the object data, as the $MKey_p^{-1}$ and thus the $OKeys$ remain the same.

4.4 Key Management

As explained in Section 4.2, the data owner generates a public master key pair $[MKey, MKey^{-1}]$ for each delegation group. After initialization through the data owner, the delegation group is responsible for $OKey$ management.

$OKeys$ are not stored by the delegates or the DGIO, rather, they are generated for every key request by partially signing the concatenation of object ID and $OKey$ version number $(ID_{O_P} + V_{OKey_{O_P}})$ stored in the DGIO. In that way only the authorized requesting client can generate the $OKey$ and cache it for future requests. Caching the keys reduces network traffic and processing load for the delegates. Because $OKeys$ are newly generated for every key request, a new $OKey_{o_p}$ is implicitly generated when $V_{OKey_{O_P}}$ changes.

4.5 Data Encryption

For data encryption in PACS three cases have to be distinguished: (1) initial data encryption on startup, when a new object is inserted into the network with access control enabled, (2) data re-encryption as a result of a privilege revocation and (3) modification of a stored data object.

Startup. Let us assume peer p wants to insert a new, protected object o into the network. At the very beginning, p has to create a delegate group, if no delegate group exists or the existing delegate group is not appropriate. To do this, p creates the master key pair $[MKey_p, MKey_p^{-1}]$.

Next, p generates the private object key $OKey_{o_p}$ for object o, by signing the ID of o (ID_{o_p}) together with a version number. As this is the start of the encryption, $V_{OKey_{o_p}} = 0$. The initial object key is therefore $OKey_{o_p} = sig_{(ID_{o_p}+0)}^{MKey_p^{-1}}$. Then p encrypts o with $OKey_{o_p}$, generating the ciphertext object o_p^{-1} as o_p^{-1}

$= enc_{o_p}^{OKey_{o_p}}$. p adds the header information $head_{o_p}$, as described in Section 3.3, to the encrypted object. Then p calculates the self-verifying identifier of o_p as $ID_{o_p}^{SV} \equiv h(head_{o_p} + o_p^{-1})$. Afterwards, p distributes $o_p^{-1} + head_{o_p}$ to the replica servers by calling $put(ID_{o_p}^{SV}, (o_p^{-1} + head_{o_p}))$ on the data storage.

What remains to be done is to build the delegate group, so that authorized peers can receive a decryption key in the absence of p. p selects n peers to do the privilege enforcement collaboratively. Next, the shares and share-shares are created and distributed to the delegates, following the procedure presented in Section 4.1. If p had to create a new delegate group, it must generate the DGIO for this new delegate group (see Section 4.3). Otherwise, the object information for o has to be added to the existing DGIO. This includes the $V_{OKey_{o_p}}$, and $enc_{ID_{o_p}^{SV}}^{OKey_{o_p}}$.

After storing the updated or created DGIO in the privilege store, peers receive the $OKey_{o_p}$ on request, if enough delegates approve the request according to the peers privileges. With the $OKey_{o_p}$, the client decrypts the $ID_{o_p}^{SV}$, then requests the data object o_p^{-1} with this ID and decrypts it.

Privilege Revocation. Data re-encryption is needed whenever a privilege on an object o is revoked by a client. A peer that possesses the privilege to grant privileges for object o to other peers has intrinsic read and write privileges on o. This is a constraint of our privilege management, because the re-encryption of an object is in fact a write operation. Before the actual revocation, the revoker requests the DGIO of object o, decrypts the $ID_{o_p}^{SV}$ with $OKey_{o_p}$ and calls $get(ID_{o_p}^{SV})$ on the data store. After receiving o_p^{-1}, r decrypts it with $OKey_{o_p}$ to get the plaintext object o. The $OKey_{o_p}$ can be requested from the delegates because the grant privilege includes read and write privileges.

Next, the revoker r revokes the privilege of peer b on object o. This operation has two parts. The first part is done by the PACS privilege management and is blanked out here. We assume that r has the appropriate privileges and therefore the removal succeeds. The second part is the generation of a new $OKey_{o_p}$ (called $OKey'_{o_p}$), followed by the encryption of o with the new $OKey'_{o_p}$. This is mandatory, since peer b may have seen and cached the $OKey_{o_p}$. Without a new $OKey'_{o_p}$ and re-encryption of o, b can still decrypt o_p^{-1} and get o in plaintext. The new $OKey'_{o_p}$ is generated by the delegates by incrementing the version counter $V_{OKey_{o_p}}$. To allow the delegates to prove the correctness of the revocation, the revoke command is first sent to the delegates for object o. Each delegate d_i that accepts the revoke returns a partial signature $sig_{(ID_{o_p}+(V_{OKey_{o_p}}+1))}^{s_i}$ privately to r. After r has received t valid partial signatures, it computes the new $OKey'_{o_p}$ as $OKey'_{o_p} = sig_{(ID_{o_p}+(V_{OKey_{o_p}}+1))}^{MKey_p^{-1}}$, following the description in Section 4.1. Only then, r submits the revoke command to the privilege store. Afterwards, the revoker r re-encrypts the data object o with the new $OKey'_{o_p}$,

$o_p'^{-1} = enc_{o_p}^{OKey'_{o_p}}$, and stores the newly encrypted object into the data storage. To do this, r computes the new self-verifying identifier $ID_{o_p}^{'SV}$ and executes $put(ID_{o_p}^{'SV}, (o_p'^{-1} + head_{o_p}))$.

Having stored the encrypted data object, the revoker informs the delegates about the update by sending the new encrypted self-verifying identifier of o ($enc_{ID_{o_p}^{'SV}}^{OKey'_{o_p}}$) to the delegates. Now the DGIO can be updated to contain the new version number $V_{OKey_{o_p}} + 1$ and the new encrypted $ID_{o_p}^{'SV}$.

The DGIO update is done either by one of the delegates or by the revoker. To make the DGIO valid again, it needs to be signed with the $MKey_p^{-1}$ key. Again, our threshold signature schema is used to generate the signature. All delegates d_i generate a partial signature of the new DGIO and send it to the coordinator (revoker r, or one of the delegates). If t of n delegates agree with the changes, the signature is complete and the DGIO can be stored in the privilege store with the valid signature.

After the re-encrypted object $o_p'^{-1}$ and the updated DGIO are stored, the outdated object o can be deleted by the revoker. The revoker sends a special *put* request $put(ID_{o_p}^{SV}, DGIO)$ to the data store, where $ID_{o_p}^{SV}$ is the self-verifying ID of the old object and DGIO is the new DGIO object. As explained in Section 3.3, this instructs the storage nodes to delete the object with ID $ID_{o_p}^{SV}$, as it is outdated.

Modifying an Object. Storing a modified object is similar to revoking a privilege. The difference is the interaction with the privilege store. When a peer p wants to update object o, it modifies the object and encrypts it with the $OKey_{o_p}$ (provided that p has already received o). Now p generates the new $ID_{o_p}^{'SV}$ and encrypts it with the $OKey_{o_p}$. This encrypted $ID_{o_p}^{'SV}$ is now sent to the delegates to request an update of the DGIO entry for object o to the new $ID_{o_p}^{'SV}$. The delegates only accept the changes to the DGIO if p has the required privileges to modify object o. If t delegates accept the changes to the DGIO and send back a partial signature of the new DGIO to p, a valid DGIO signature is created. The modified data object can now be stored in the data store using $put(ID_{o_p}^{'SV}, (o_p^{-1} + head_{o_p}))$. After the updated and signed DGIO is put into the privilege store, the old version of o can be deleted by executing $put(ID_{o_p}^{SV}, DGIO)$.

5 Complexity Analysis

Here, we evaluate the performance of PACS with client-side enforcement. We compare PACS with the pure client-side enforcement abbreviated as "client" and pure delegate enforcement shortened as "delegate", as introduced in Section 2. For the sake of brevity, we only compare the simple approaches and do not take into account any optimizations. We also abstract from the details of data encryption (Section 4.5) and delegation group management (Section 4.3) to get a balanced and comprehensible comparison.

Table 1. Number of Messages for the Different Enforcement Approaches

	Cold Request	Hot Request	Insert	Revoke	Grant
Client	$2 * log(n)$	$log(n)$	$2 * log(n)$	$4 * log(n)$	$log(n)$
Delegate	$(t+1) * log(n) + 2t$	$t * log(n) + 2t$	$2 * log(n)$	0	0
PACS	$2 * log(n) + 2t$	$log(n)$	$2 * log(n)$	$4 * log(n) + 2t$	0

Table 2. Amount of Data to be Transferred in the Different Enforcement Approaches

	Cold Request	Hot Request	Insert	Revoke	Grant
Client	$KF + o$	o	$KF + o$	$2KF + 2o$	KF
Delegate	$DGIO + 2t * o$	$2t * o$	$DGIO + o$	0	0
PACS	$DGIO + t * sig + o$	o	$DGIO + o$	$2 * (DGIO + t * sig + o)$	0

Table 3. Computational Effort for the Different Enforcement Approaches

	Cold Request	Hot Request	Insert	Revoke	Grant
Client	$2 * dec$	dec	enc	$2 * dec + enc$	enc
Delegate	$t * enc + 2t * dec$	$t * enc + 2t * dec$	enc	0	0
PACS	$t*sig+t*enc+(t+1)*dec$	dec	enc	$2t*sig+(2t+1)*$ $enc+(2t+1)*dec$	0

We estimate the number of messages needed, the amount of data that has to be transferred and the computational power needed by the participants to perform the enforcement. The effort for privilege management is not cosidered here. For simplicity, we assume that all data is stored in a DHT with the number of messages used to retrieve data being $log(n)$ where n is the network size. t is the number of delegates that need to be contacted, o is the data object and sig a signature. We abstract here from the overhead of secure communication between the participants as long as it is the same for all approaches. In addition, we assume that the DGIO has the same size as the key file (KF) in the client case. Similarly, the effort to create or modify a DGIO or a KF are the same. We distinguish between decryption (dec), encryption (enc) and signature generation (sig).

We compare the three approaches for an access, an insert, an update, a revoke and a grant request. We distinguish between the first (cold) request and the following requests (hot), as client and PACS enforcement use key caching. The results shown in Tables 1, 2 and 3 and Figures 3 and 4 are explained in the following.

Cold Request. In the client enforcement approach, a DHT request retrieves the key and another DHT request retrieves the data object. DGIO and the object need a decrypt operation. In PACS we need to retrieve the DGIO. Then at least t delegates are contacted to create and return the partial signatures.

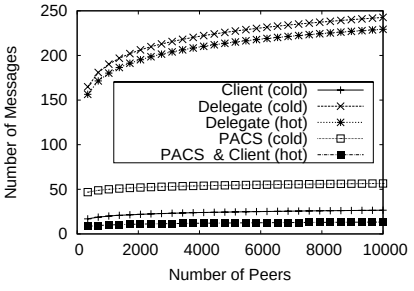


Fig. 3. Number of Messages for a Request with $t = 15$ and $log_2(n)$

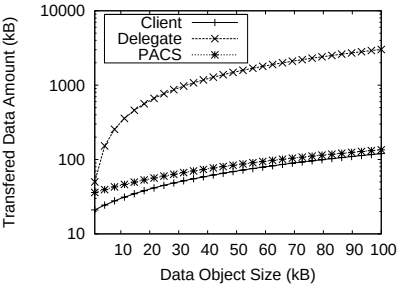


Fig. 4. Amount of Transferred Data for a Cold Request (sig = 1 kB, DGIO = 20 kB, $t = 15$, $log_2(n)$)

The retrieval of the data object results in an additional DHT request. In the delegation approach we retrieve a kind of DGIO identifying the delegates for the object. Afterwards, at least t delegates are asked to return partial decrypted data objects securely (i.e. encrypted).

Hot Request. Here we consider only the second request by a client for an object. If the privileges have not changed (if no revoke occurred), the keys cached by the client and PACS can be used again.

Insert. An insert of a new object into an established access control mechanism triggers an update or creation of a KF/DGIO (DHT request). In addition the data has to be encrypted and stored.

Update. In all approaches the altered object is encrypted and afterwards inserted into the network, which causes $log(n)$ messages. As there is no difference between the approaches, this is omitted.

Revoke. In the PACS and client approach, privilege revocation results in requesting and re-encrypting the object. In the client approach, this corresponds to the request effort plus the insert effort. In PACS (if the revoker is not the data owner) t delegates need to be asked for a new key. This results in additional $2t$ messages. In the delegate approach there is no effort, as no one (beside the data owner) has the decryption key in plain text, so no re-encryption is needed.

Grant. Granting a privilege causes an update of the key file in the client approach, which results in $log(n)$ messages. The PACS and delegate approach require no effort.

Figure 3 illustrates the predicted number of messages for various approaches in networks with up to 10,000 nodes. PACS is clearly superior to the delegate approach and when the cache is warm, it should perform as well as the client approach. Figure 4 shows the expected sizes of transferred data in kB. The volume load in the network in PACS is much lower than in the delegate method, and similar to the client enforcement method.

6 Discussion and Future Work

Client-side enforcement in PACS is a compromise between performance and power of the access control system. As can be seen from the complexity estimations (Figures 3 and 4), the pure delegate approach is too expensive in the amount of transferred data and communication cost. Especially the fact that each object has to be tranmitted t times is responsible for the limited scalability, as the communication costs linearly increase with the number of requests. The best performance is achieved by client-side enforcement. However, this has limited access control power. In fact, the pure client-side enforcement shown here can only restrict read access to data objects and offers no administrative distribution, grouping of privileges, roles, etc. as provided in PACS. PACS pays for its additional access control power with every cold request but it performs better than the pure delegation approach in terms of communication costs and the size of data that is transmitted. PACS needs as much computational power as the pure delegate approach, but this is not that critical, as partial signature generation is done in parallel and every peer generates only one signature per request.

All the cryptographic primitives used in PACS have been proven secure elsewhere. As the combination of methods used in PACS does not release or utilize any additional information, also the combination of the cryptographic methods is secure. In such a dynamic environment one has to find tradeoffs between security and performance, and between the power of the access control and its generated overhead. In that way PACS is a compromise in both respects. It guarantees the security of data while producing acceptable additional effort. Furthermore, we get a powerful access control enforcement mechanism with minor restrictions, but with an acceptable overhead.

In the future, we plan to make several extentions to our work. First of all, PACS can be employed on an unreliable P2P network. For non-crucial data, especially data that can disappear for a certain amount of time, this may be less expensive and less complicated. Alternatively, we could dispense with the privilege store. The task and responsibility of storing the privileges reliably in the network can be handed over to the delegates that have to enforce the privileges anyway. In this way, PACS would become more lightweight, which would lead to an easier integration into existing applications based on P2P networks. Furthermore, we want to implement this extention in our PACS prototype and run comparative performance studies.

7 Conclusions

We presented an extention of PACS that enables the replication of confidential data and distribution of privilege enforcement. Our privilege enforcement mechanism is based on threshold encryption / signature and private key encryption with key distribution. The chosen combination enables PACS to enforce powerful access control models like RBAC with administrative distribution, which is

impossible with the currently known approaches. Regarding the gained flexibility, the additional overhead of PACS is acceptable. With PACS, data in a P2P network can be protected almost as flexibly as in a centralized scenario. This may open up P2P networks for further application areas.

References

1. Halevy, A.Y., et al.: Schema Mediation in Peer Data Management Systems. In: ICDE, pp. 505–516 (2003)
2. Sturm, C.: Orchestrating Access Control in Peer Data Management Systems. In: Grust, T., Höpfner, H., Illarramendi, A., Jablonski, S., Mesiti, M., Müller, S., Patranjan, P.-L., Sattler, K.-U., Spiliopoulou, M., Wijsen, J. (eds.) EDBT 2006. LNCS, vol. 4254, pp. 66–74. Springer, Heidelberg (2006)
3. Sturm, C., Dittrich, K., Ziegler, P.: An Access Control Mechanism for P2P Collaborations. In: DAMAP (2008)
4. Miklau, G., Suciu, D.: Controlling Access to Published Data Using Cryptography. In: VLDB, pp. 898–909 (2003)
5. Bouganim, L., et al.: Client-Based Access Control Management for XML Documents. In: VLDB, pp. 84–95 (2004)
6. Desmedt, Y.G.: Threshold Cryptography. European Transactions on Telecommunications and Related Technologies 5(4), 449–457 (1994)
7. NIST: Role Based Access Control. ANSI INCITS 359-2004 (February 2004)
8. Crispo, B., et al.: P-Hera: Scalable Fine-grained Access Control for P2P Infrastructures. In: ICPADS, pp. 585–591 (2005)
9. Berket, K., Essiari, A., Muratas, A.: PKI-Based Security for Peer-to-Peer Information Sharing. In: P2P, pp. 45–52 (2004)
10. Goh, E.J., et al.: SiRiUS: Securing Remote Untrusted Storage. In: Proceedings of the Network and Distributed System Security Symposium, NDSS 2003 (2003)
11. Koç, E.: Access Control in Peer-to-Peer Storage Systems. Master's thesis, ETH Zurich (October 2006)
12. Koç, E., et al.: PACISSO: P2P Access Control Incorporating Scalability and Self-Organization for Storage Systems. Technical Report TR-2007-165, Sun Microsystems, Inc. (June 2007)
13. Sandhu, R., Zhang, X.: Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. In: SACMAT, pp. 147–158 (2005)
14. Saxena, N., et al.: Threshold Cryptography in P2P and MANETs: The Case of Access Control. Comput. Netw. 51(12), 3632–3649 (2007)
15. Naor, M., Wool, A.: Access Control and Signatures via Quorum Secret Sharing. IEEE Trans. Parallel Distrib. Syst. 9(9), 909–922 (1998)
16. Castro, M., et al.: Secure Routing for Structured Peer-to-Peer Overlay Networks. SIGOPS Oper. Syst. Rev. 36(SI), 299–314 (2002)
17. Committee, O.A.C.T.: eXtensible Access Control Markup Language (XACML) Version 2.0 (2005), `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`
18. Stoica, I., et al.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In: SIGCOMM 2001, pp. 149–160 (2001)
19. Desmedt, Y.G., Frankel, Y.: Threshold Cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
20. Daemen, J., Rijmen, V.: The Design of Rijndael. Springer, New York (2002)

21. Rivest, R.L., et al.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Commun. ACM 21(2), 120–126 (1978)
22. Shamir, A.: How to Share a Secret. Commun. ACM 22(11), 612–613 (1979)
23. Desmedt, Y., Frankel, Y.: Shared Generation of Authenticators and Signatures. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 457–469. Springer, Heidelberg (1992)
24. Gennaro, R., et al.: Robust and Efficient Sharing of RSA Functions. Journal of Cryptology 13(2), 273–300 (2000)
25. Herzberg, A., et al.: Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995)

Spatiotemporal Access Control Enforcement under Uncertain Location Estimates

Heechang Shin and Vijayalakshmi Atluri

MSIS Department and CIMIC, Rutgers University, USA
{hshin,atluri}@cimic.rutgers.edu

Abstract. In a mobile environment, user's physical location plays an important role in determining access to resources. However, because current moving object databases do not keep the exact location of the moving objects, but rather maintain their approximate location for reasons of minimizing the updates, the access request evaluation cannot always guarantee the intended access control policy requirements. This may be risky to the system's security, especially for the highly sensitive resources. In this paper, we introduce an authorization model that takes the uncertainty of location measures into consideration for specifying and evaluating access control policies. An access request is granted only if the confidence level of the location predicate exceeds the predefined uncertainty threshold level specified in the policy. However, this access request evaluation is computationally expensive as it requires to evaluate a location predicate condition and may also require evaluating the entire moving object database. For reducing the cost of evaluation, in this paper, we compute lower and upper bounds (R_{min} and R_{max}) on the region that minimize the region to be evaluated thereby allowing unneeded moving objects to be discarded from evaluation. We show how R_{min} and R_{max} can be computed and maintained, and provide algorithms to process access requests.

1 Introduction

Unlike the traditional access control system, in a mobile environment, a person's physical location plays an important role in determining access rights. For example, access to certain important resources of an organization can be restricted to employees who are currently located within the office area. This concept of location-based access control (LBAC) system is not new. For example, in [6], when an access request is made from a moving object, the system checks whether the requester lies within the authorized region and only if this is the case, access is granted.

We can categorize access control rules based on the mobility of subject/resources as (i) *static subjects to access mobile resources* (SM), (ii) *mobile subjects to access static resources* (MS), and (iii) *mobile subjects to access mobile resources* (MM). First, SM is to restrict access to resources based on the spatiotemporal locations of resources. For example, the New York branch operations department of a truck company are allowed to track the locations of dispatched trucks that are currently located within New York City. Second, MS restricts access to static resources based on the spatiotemporal location of the subjects. For example, security managers are allowed to read or write the

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 159–174, 2009.

mobile network data only when they are currently located within the server farm room. Finally, in MM, access control decisions are made based on the spatiotemporal location of subjects and resources. For example, all supervisors who are currently located within the office building can locate their employees only when they are also currently in the same building.

The proposed LBAC systems [4, 6, 9, 18] assume that provided location measure of moving object is always accurate. For example, in GEO-RBAC [9], positions can be real or logical: the real position corresponds to the position on the Earth of the mobile user obtained from location-sensing technologies such as Global Positioning Systems (GPS) while the logical position is modeled as a polygon feature such as city, i.e., a real position acquired through GPS can be mapped to a corresponding road segment (logical position). A spatial role is activated based on the location (either logical or real) of the user. However, considering the fact that users are moving objects, most of the time, the provided location information is not precise because of continuous motion [21]. In fact, most of currently proposed moving object databases do not keep the exact location of the moving objects, rather maintain the approximate value of the location in order to minimize the updates. Therefore, a location measure stored in the moving object database should be modeled as a region instead. In general, we call this inherent error of a location measure as *location uncertainty*. However, if we consider the inherent uncertainty of location measures, the role activation in GEO-RBAC cannot guarantee the desired security. In other words, their underlying assumption that any logical position can be computed from real positions by using specific mapping functions are not true any longer because it is possible that several logical positions can be mapped from a single real position. This may incur huge risks to the security of the system especially for highly sensitive resources. Therefore, it is essential that all LBAC systems must incorporate the concept of uncertainty within the model.

To the best of our knowledge, the work of Ardagna et al. [3] is the only LBAC model where uncertainty is considered. They present a model for representing and evaluating a set of location predicates. Each access request can gain access to the specified resources only if the confidence level of the location predicate result exceeds the predefined uncertainty threshold level. Formally, given an access control rule's location predicate and a user o, we need to evaluate the probability p_o, the chance that o satisfies the given location predicate, to determine the satisfiability of the predicate (i.e, o is located within an authorized region R). Given an authorized region R and o's uncertainty region denoted as $o.ur$, p_o is computed as

$$p_o = \int_{o.ur \cap R} f_o(x)dx \tag{1}$$

where x is the location of o in d-dimensional data space D, f_o is the probability density function (PDF), and $o.ur \cap R$ is the intersection of $o.ur$ and R. In other words, p_o is the confidence level of the location predicate result. The user o gains access to the resources only if $p_o \geq p_c$ where p_c is the predetermined predicate threshold. However, their model has the following limitations: (i) the uncertainty thresholds for location predicates are globally fixed values, thus lacking the specification power for different situations: for example, the minimum threshold of location predicate for granting access is a globally fixed level, and therefore, it cannot differentiate between highly security-sensitive area

and less sensitive area by assigning different confidence levels; and (ii) resources are assumed to be always static, and therefore, only MS type of security policies for a mobile environment can be evaluated.

To address the above limitations, we introduce an access control model that embeds uncertainty within the model. In this model, we allow varying threshold levels of location predicates; thus, it is possible to differentiate the highly security sensitive area and less sensitive area in an access control rule. Also, in addition to MS type of security policies, SM and MS type of policies are supported. Although Ardagna et al.'s model can be extended to support MM and SM type of queries by incorporating location predicates in specifying resources, the main challenge of supporting them in their model is the evaluation part, which is the focus of this paper: it is hard to evaluate the resources' satisfiability to an access request. For example, if location of employees in a given area are requested, considering the location uncertainty, the access control enforcement system cannot simply release the location of employees inside the region since it is still possible that some people who are believed to be located outside may, in fact, be inside, and vice versa, i.e., people believed to be inside may actually be outside. Under Ardagna et al.'s model, in order to guarantee the correctness of the query results, it may require to evaluate all the moving objects in the database since they only allow Boolean queries.

Our main objective in this paper is to reduce the cost of location predicate evaluation. There are two main challenges: (i) the computation of Equation (1) is computationally expensive. For example, under the normal distribution case, $o.ur \cap R$ is not symmetric with respect to the mean [20], and therefore, it is expensive to compute; (ii) the size of uncertainty region grows as time elapses because the actual location can deviate further than the measured one. This implies that location predicate evaluation cannot be computed in advance. In order to address the first issue, Yufei et al. [20] propose a Monte Carlo based approach. This method generates inputs randomly, performs a deterministic computation using the inputs and aggregates the results of the individual computations into the final result. But it is relatively accurate only if the sampled points are sufficiently large. This is because, the computation of probability is based on the sampling, the result of this approach is close to the actual value only if there are enough number of samples (i.e., at the order of 10^6 in their experimental study). Even worse when considering the second problem, in the moving object database, Equation (1) needs to be computed within the reasonable amount of time because the satisfying condition of location predicate changes as the position of o is constantly updated. Also, because the size of uncertainty region grows as time elapses after the last location update, it requires the continuous evaluation of the specified location predicates. Therefore, we need to reduce the cost of computation of p_o as much as possible.

Our proposed approach is to find the upper and lower bounds of the region to be evaluated, essentially identifying two regions: (i) the first, called R_{min}, is the region that guarantees the correctness of the location predicate evaluation if the location estimate is within this region, and (ii) the second, called R_{max}, is the region where any location measure outside of this region is guaranteed to have no probability to satisfy the given predicate. Once these regions are found, the cost of location predicate evaluation process is significantly reduced because it requires simple location containment

test to evaluate the predicate correctness for most of location measures instead of expensive computation of Equation (1). More specifically, instead of computing p_o by using Equation (1) for all the location measures, we take the following steps:

1. Those objects which are located outside of R_{max} are filtered out from the candidate set for examining their satisfaction for the given predicate;
2. Among the objects located within R_{max}, we do not need to evaluate p_o of those objects located within R_{min} because it is guaranteed to have $p_o \geq p_c$;
3. p_o needs to be computed only for those objects located in $R_{max} - R_{min}$. In order to minimize the cost of p_o evaluation, our objective is to minimize the area size of $R_{max} - R_{min}$.

We discuss how to generate R_{max} and R_{min} while minimizing $R_{max} - R_{min}$. Specific probability distribution such as uniform distribution is considered to find R_{max} and R_{min}.

The rest of the paper is organized as follows. Section 2 introduces preliminaries of the paper. In Section 3, we present an access control model where the concept of uncertainty is embedded. Section 4 introduces our novel concept of R_{max} and R_{min} under specific probability distribution such as uniform distribution and presents our algorithm to handle imprecise access requests. Related work is presented in Section 5. Section 6 concludes the paper with some suggestions for future work.

2 Preliminaries

In this section, we describe the uncertainty model of moving objects and present an overview of our system architecture in a mobile environment.

2.1 Uncertainty of Moving Objects

According to [13], in the mobile network environment, no technology is available that ensures precisely the exact user locations. Thus, a position of a moving object, instead of a single location point, is rather specified with a range, called uncertainty region. The uncertainty is caused by the sampling error and the measurement error [17].

Sampling Error. It is unrealistic to obtain the current location of the moving objects continuously under the existing location sensing technologies and database technologies, and the position is collected at discrete instances of time such as every few seconds instead [17]. The solid line in Figure 1 represents the projected movement of a moving object in one dimensional space (x axis) and time (t axis). Linear interpolation is used to project positions between two consecutive location updates: the sampled positions become the end points of single line segments, and the entire polyline (i.e., solid line) represents the projected movement of the object. However, this approach brings the error due to the position estimation methods of moving objects within any single line segment except the end point. For example, in Figure 1, the dashed line shows the actual locations of the object between t_0 and t_5. After the location is updated, because the position of the moving object is unknown until the next location update, the actual location can be anywhere within the so called uncertain region.

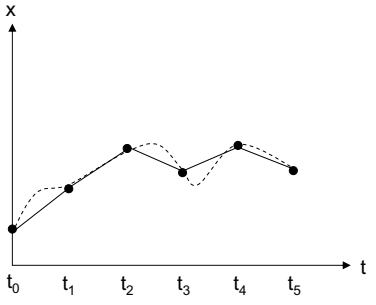


Fig. 1. Position History

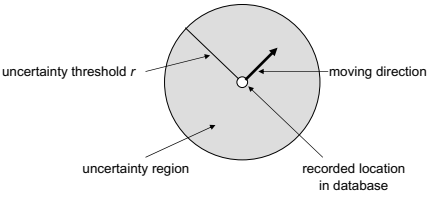


Fig. 2. Uncertain Object Example

Measurement Error. Location sensing techniques determine the accuracy and the quality of the location measurements. Pfoser et al. [17] propose that given a location measure $x = (\mu_{x_1}, \mu_{x_2})$ in 2-dimensional space, the error in a positional GPS measurement can be described by the bivariate normal distribution where the mean (or variance) of the distribution is the measured location of the coordinate system (or $\sigma = \sigma_{x_1} = \sigma_{x_2}$), and the covariance is 0. More specifically, a positional GPS measurement is described by

$$f_o(x) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x_1-\mu_{x_1})^2+(x_2-\mu_{x_2})^2}{2\sigma^2}} \tag{2}$$

This implies that the distribution in the x_1-x_2 plane is circular, and within the range of $\pm\sigma$ of the mean, 39.35% of the probability is concentrated. Depending on the location sensing techniques, the value of σ can be determined. Given $\sigma_1 < \sigma_2$ with the same location measure, it is obvious that the location measure involved with σ_2 would involve more measurement error. Liu et al. [15] use two performance metrics for measurement error:

- *Accuracy* is the degree of closeness between the estimated location and the true location.
- *Precision* is the distribution error between the estimated location and the true location. In other words, it measures how consistently the system measures the given location.

For example, HORUS [26, 24] has the accuracy of 2m and the precision of 90% within 2.1m [15], and the maximum measurement error level can be specified as 2m + 2.1m = 4.1m with 90% certainty.

Due to the above errors, the system does not have users' precise positions, and the user can be anywhere in an uncertainty region.

Definition 1 (Moving Object Uncertainty). *Given a set of moving objects O, uncertainty of a moving object $o \in O$ in the d-dimensional data space D is conceptually described by (i) a d-dimensional uncertainty region, denoted by $o.ur$ and (ii) a PDF f, denoted by $f_o(x)$ where $x \in D$ is the d dimensional location.*

Notice that $f(x) \geq 0$ for any point $x \in D$ and $\int_{o.ur} f_o(x)dx = 1$. Also, $f_o(x) = 0$ if o is located outside of $o.ur$. We use $loc(o)$ to denote the last update location of o in the database. Uncertainty region is often represented with a circle with uncertainty threshold r, which is determined by

$$r = m_{error} + v_{max} \cdot |t_c - t_u| \tag{3}$$

where m_{error} is the measurement error level of location sensing technologies, v_{max} is the maximum speed, t_c is the current time, and t_u is the last update time. In other words, the circle with radius r is the region that a user can be possibly located after the last location update. Figure 2 illustrates such an example. Observe that m_{error} refers to the measurement error and $v_{max} \cdot |t_c - t_u|$ refers to the sampling error. Since the PDF of an uncertain object is often unavailable explicitly, a set of samples are drawn or collected in the hope of approximating the PDF [16]. However, in some applications, it would make sense to use the specific PDF f for specifying the uncertainty region. For example, Wolfson et al. [19] propose that the object location follows the Gaussian distribution over the uncertainty region. Also, uniform distribution is used in many application scenarios [17, 21] to represent an uncertainty region.

2.2 System Architecture Overview

We assume the system in a mobile environment comprises of the following components:

Location Server (LS). The main objective of LS is to enforce access control policies in order to protect the important resources. Location information can be maintained by either LS itself or another trusted third party such as a mobile service provider. The current location of moving objects are stored and updated accordingly in order to provide most up-to-date location information to a service requester. If LS maintains the location information, users' mobile devices directly provide such information via wireless communication periodically, the installed sensors can approximate it. For example, the Active Badge [1] detects the location of each user in a building. Each individual carries a device called, badge, which is associated with the identifier of the user. A building is equipped with sensors detecting positions of badges. A person's location is determined by using an active badge which emits a signal every 15 seconds.

Requester. A requester subscribes to a service in order to gain access to the resources. In a mobile environment, there are two types of resources that a requester can gain access to: static resources (e.g. repository room or printer) and mobile resources (location of vehicles). For example, consider a work environment where all the documents can only be accessed by employees only while they are physically located in the office. Similarly, requester can be either static or mobile. For example, when a mobile requester submits an access request to the documents in the repository, LS checks the physical location of the requester, and only if the subject is within premises of the office, he is given an access. However, in some cases, the location of requestors is always fixed. For example, an emergency message (e.g., reverse 911) need to be delivered to students' mobile devices by the university police office, only while the students' are on campus.

Under our framework, LS is responsible for enforcing security policies. More specifically, when a user (mobile or static) submits an access request, the LS searches relevant security policies that are applicable to the submitted access request. If a location predicate is included in a relevant policy, location information is either provided by the LS (if the location information maintained by the LS) or retrieved from the trusted location service provider. The provided location measures should compute the inherent uncertainty of provided location measures (discussed in Section 2.1). After evaluating the policies, the LS returns answers to the requester.

3 Location-Based Access Control Model under Uncertain Location Estimates

In this section, we introduce a location-based access control model suitable for moving object data with inherent uncertainty by extending the GSAM [4, 5]. An access control rule, in general, is specified on the basis of three parameters, $\langle s, o, p\rangle$ which states that s is authorized to exercise privilege p on resource o[1]. However, this basic access control rule lacks specification power to include moving object data since an access control rule should be capable of specifications based on spatiotemporal attributes of both subjects and resources that are functions of time. In the following, we extend the basic authorization to accommodate this.

Definition 2 (Access Control Rule). *An access control rule is a triple of the form $\langle$se, re, pr$\rangle$ where se is a subject expression that denotes a set of authorized subjects, re is a resource expression that denotes a set of authorized resources, and pr is a set of privilege modes that denotes the set of allowed operations.*

In this paper, we use $\mathcal{P}$ to denote the set of access control rules stored in the LS. Given an access control rule $\alpha \in \mathcal{P}$, *se(α), re(α) and pr(α)* denote the set of subjects satisfying subject expression, the set of resources satisfying resource expression, and the set of privileges, respectively, of α. Also, $\alpha(R_r)$ and $\alpha(R_s)$ denote the authorized region specified in se(α) and re(α), respectively. In the following, we discuss these concepts in detail.

Definition 3 (Subject Expression). *A subject expression is a triple of the form $\langle$R, sc, ue$\rangle$ where R is the role to which the subject belongs, sc is the location predicate, called* scene*, which can be associated with a set of geospatial and temporal extents, and ue is an uncertainty expression associated with a* scene*.*

Similar to subject expression, a resource expression includes (i) an object type t which evaluates the membership of the object in categories, or values of properties on metadata, and (ii) location predicate with uncertainty expression.

Definition 4 (Resource Expression). *A resource expression is a triple of the form $\langle$t, sc, ue$\rangle$ where t the object type to which the resource belongs, sc is a location predicate, called scene, which can be associated with a set of geospatial and temporal extents, and ue is an uncertainty expression associated with a scene.*

[1] 'Object' is more general term to specify o, but in order not to confuse with moving objects, we specify o as resources throughout the paper.

In this paper, we assume the formalism developed in [5] to specify *R* and *sc* in a subject expression. Due to space limitations, we do not review the details. In short, *R* refers to a role in RBAC with roles organized as a hierarchy, and *sc* is a conceptual event or region that can be mapped to a set of bounding boxes represented with $\langle label, lt, lg, h, w, [t_b, t_e]\rangle$ where *label* is a descriptive scene name, $\langle lt, lg, h, w\rangle$ denotes latitude, longitude, height and width of a bounding box covering a geographic area of the *scene* during temporal period between t_b and t_e[2]. An object type o in an object expression can be organized into a hierarchy similar to a role hierarchy. In [5], only geospatial objects are considered, but it can be extended to support other types of objects as well. Both subject expression and object expression can also contain more complex static predicates or expressions other than roles if necessary.

An uncertainty expression *ue* is a logical expression denoting uncertainty level of the corresponding *scene* in both se and re. As we have discussed in Section 2.1, any location measure stored in the database includes inherent uncertainty.

Definition 5 (Uncertainty Expression). *Given $\alpha \in \mathcal{P}$ and a finite set of scenes $S = \{sc_1, sc_2, \cdots, sc_m\}$ used in se(α) or re(α), an uncertainty expression, denoted as* ue*(α), is defined as follows:*

- *If sc_i is a scene, op $\in \{=, \neq, <, >, \leq, \geq\}$, and p_c is a real number in the range from 0 to 1, sc_i op p_c is a ue.*
- *If ue_1 and ue_2 are two uncertainty expressions, $ue_1 \wedge ue_2$, $ue_1 \vee ue_2$, $\neg ue_1$, and (ue_1) are ue.*

Although we allow any logical operator (i.e, $=, \neq, <, >, \leq, \geq$) for specifying *ue*, we particularly focus on $\geq$ operator in this paper since it plays an important role to prune out moving objects (either subjects or resources) that do not satisfy the uncertainty threshold specification (i.e., p_c). Evaluation of *scene* (location predicate) results in the following form [*result_set, timeout*] stating that each element in the *result_set* includes a moving object and its corresponding confidence level, and every element in the *result_set* exceeds the specified uncertainty threshold level in the corresponding uncertainty expression. More specifically, the confidence value of each object o, denoted as p_o, is compared with the predetermined value of threshold in *ue*, denoted as p_c, and those objects whose confidence level is greater than the threshold are included in the *result_set*. Although [3] requires two thresholds for accepting or rejecting the evaluation, the request is granted only if the confidence level is above the upper threshold. Therefore, without loss of generality, we only require one threshold for evaluating location predicates. Also, the timeout represents the time validity of the result. This timeout takes into account that location values may change rapidly, even during policy evaluation. After it is expired, *scene* must be reevaluated to guarantee the correctness of the evaluation since the location measures are constantly updated.

[2] Actually, *sc* corresponds to the inarea() location predicate in [2]. In this paper, we mainly focus on this type of predicate evaluation because other location predicates introduced in [2] is a special case of the proposed approach. For example, in case of velocity, it is the special case of the proposed approach with one dimensional space, and thus, the proposed approach is general enough to evaluate other location predicates.

- $\alpha_1 = \langle\{\text{operations\_dept}(x)\}, \{\text{truck}(y), \text{New\_York\_City}(y), \text{New\_York\_City} \geq 0.7\}, \{\text{track}\}\rangle$: This rule states that the operations department can track the locations of dispatched trucks that are currently located within New York City with greater than 70 % confidence.
- $\alpha_2 = \langle\{\text{Security\_manager}(x), \text{server\_farm\_room}(x), \text{server\_farm\_room} = 1.0\}, \{\text{mobile\_network}(y)\}, \{\text{read} \wedge \text{write }\}\rangle$: This rules states that all security mangers who are currently located within the server farm room with confidence level of 100% can read or write the mobile network data.
- $\alpha_3 = \langle\{\text{Supervisor}(x), \text{office\_building}(x), \text{office\_building} \geq 0.8\}, \{\text{employee}(y), \text{office\_building}(y), \text{office\_building} \geq 0.9\}, \{\text{locate}\}\ \rangle$: This rules states that all supervisors who are currently located within the office building with confidence level greater than 80 % can locate their employees who are also currently located within the building with greater than 90 % confidence level.

The access control rules α_1, α_2, α_3 refer to SM, MS, and MM type respectively. Each access control rule has its own uncertainty level specified for location predicates. We consider that the network configuration in a server farm room in α_2 is the most highly sensitive to security because configuration must be performed according to the highest security standards. Therefore, 100% of location confidence should be guaranteed to do such a job. Accessing the locations of employees in the office building is considered less critical but still to be handled in a highly secured environment and to be granted only to selected personnel, according to the laws and regulations in force [2]. Finally, we consider tracking dispatched trucks for operational purposes as the lowest critical to security.

4 Security Policy Evaluation of Uncertain Location Samples

In this section, we present our proposed approach that evaluates access control policies efficiently. Then, our solutions considering certain reasonable assumptions are described.

4.1 Proposed Approach

Our approach is to find two regions: (i) the first region, called R_{min}, is the region that guarantees the correctness of the location predicate evaluation if the location estimate is located within this region, and (ii) the second region, called R_{max}, is the region where any location measure located outside of this region is guaranteed to have no probability to satisfy the given predicate. Formally, given an authorized region $R = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d] \subseteq D$ where $a_i < b_i$ for $i = 1, 2, \cdots, d$, we want to find $c_{in}, c_{out} \geq 0$ such that $R_{min} := [a_1 + c_{in}, b_1 - c_{in}] \times [a_2 + c_{in}, b_2 - c_{in}] \times \cdots \times [a_d + c_{in}, b_d - c_{in}]$ ($a_i + c_{in} < b_i - c_{in}$ for $i = 1, 2, \cdots, d$) and $R_{max} := [a_1 - c_{out}, b_1 + c_{out}] \times [a_2 - c_{out}, b_2 + c_{out}] \times \cdots \times [a_d - c_{out}, b_d + c_{out}]$ ($a_i - c_{out} < b_i + c_{out}$ for $i = 1, 2, \cdots, d$) where $\forall o \in O$, $\min(p_o) \geq p_c$ if loc(o) is contained in R_{min} and $\max(p_o) \leq p_c$ if loc(o) is contained in $D - R_{max}$.

Figure 3 illustrates R_{min} and R_{max}. In case of R_{min}, the original authorized region R is reduced to R_{min} by c_{in} in every dimension so that if any location measure loc(o)

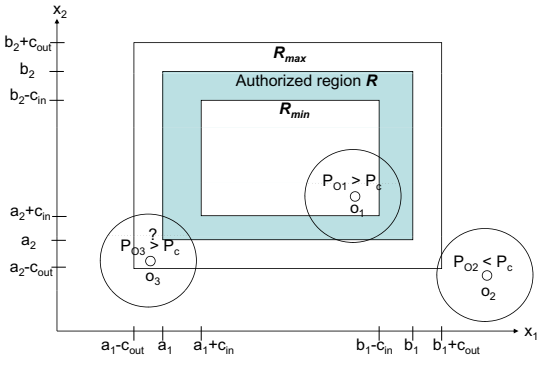


Fig. 3. Use of R_{min} and R_{max}

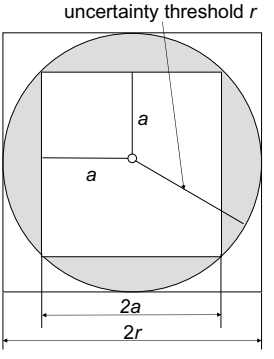


Fig. 4. Approximation of Uncertainty Region

that is stored in the last update is contained within R_{min}, we can guarantee that $p_o \geq p_c$. Therefore, in case of o_1, it is guaranteed to have $p_{o_1} \geq p_c$ because loc(o_1) is located within R_{min}. Similarly, any location measure loc(o) outside of R_{max} is guaranteed to satisfy $p_o < p_c$. For example, o_2 is located outside of R_{max}, i.e., loc(o_2)is contained in $D - R_{max}$, and thus, $p_{o_2} < p_c$ holds. However, we do not know how the result of location predicate evaluation for any location measure located in $R_{max} - R_{min}$. Therefore, in this case, we have to manually compute p_o for any $o \in O$ located within this region, i.e., we should evaluate p_{o_3} in order to see if $p_{o_3} \geq p_c$ holds. This example illustrates that it is important to have $R_{max} - R_{min}$ as small as possible. In other words, our objective is to compute the minimized value of c_{in} and c_{out} and therefore, the number of computations for Equation (1) is minimized as well.

Obviously, the main benefit of our proposed approach is that the cost of location predicate evaluation process is significantly reduced once R_{min} and R_{max} are computed because it requires simple location containment test to evaluate the predicate's correctness for most of location measures instead of expensive computation of Equation (1). Throughout the paper, we restrict our discussion on 2-dimensional space for its easiness to illustration. However, it is simple to extend to a higher dimensional space.

4.2 Uniform Distribution Case

Here we discuss how to compute the value of c_{in} and c_{out} to find corresponding R_{min} and R_{max} when uncertainty region is approximated with square under the assumption of uniform distribution.

Approximation of Uncertainty Region. Suppose a security policy is evaluated over a moving object $o \in O$ in 2-dimensional data space D. For example, o is one of the resources (or subjects) in SM (or MS) type of policies while o can be both resources and subjects in MM. Also, an authorized region R is specified in either se or re or both. In case of a circular shape of $o.ur$ centered at (μ_{x_1}, μ_{x_2}) with radius r $(\mu_{x_1}, \mu_{x_2} \geq 0)$, its PDF $f_o(x_1, y_1)$ $(x_1, y_1 \geq 0)$ is defined as

$$f(x_1, x_2) = \begin{cases} g(x_1, x_2) & \text{if} (x_1 - \mu_{x_1})^2 + (x_2 - \mu_{x_2})^2 \leq r^2 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $g(x_1, x_2)$ is a probability distribution such as uniform or bivariate normal distribution. Then, p_o is defined as

$$p_o = \int_{\max(a_1, \mu_{x_1} - r)}^{\min(b_1, \mu_{x_1} + r)} \int_{\max(a_2, \mu_{x_2} - \sqrt{r^2 - (x_1 - \mu_{x_1})^2})}^{min(b_2, \mu_{x_2} + \sqrt{r^2 - (x_1 - \mu_{x_1})^2})} f(x_1, x_2)\, dx_2\, dx_1 \quad (5)$$

However, it turns out that even with the uniform distribution, the evaluation of Equation (5) is very expensive. We can use square instead of circle for representing an uncertainty region for computing R_{min} and R_{max}. Figure 4 illustrates that we use two squares to approximate the circular shape of uncertainty region. For example, in case of R_{min}, we can use a rectangle whose circumcircle passes through all the vertices of it: the inner square with edge's size$= 2a$ where $a = \frac{r}{\sqrt{2}}$ because the radius of the circumcircle is the same as the radius of the polygon as shown in the figure. This is because we want to have the minimum value of p_o for objects located within R_{min} satisfy $p_o \geq p_c$. Then, the size of area($R \cap o.ur$) is getting smaller, implying that p_o is getting smaller compared to the original value of p_o. Given an uncertainty region $o.ur$ with circle shape where the center is (μ_{x_1}, μ_{x_2}) and the radius of r, the corresponding rectangle becomes $[\mu_{x_1} - \frac{r}{\sqrt{2}}, \mu_{x_1} + \frac{r}{\sqrt{2}}] \times [\mu_{x_2} - \frac{r}{\sqrt{2}}, \mu_{x_2} + \frac{r}{\sqrt{2}}]$.

In case of R_{max}, we want to have $\forall o \in O\ p_o \leq p_c$ satisfied. Thus, we want to find a rectangle whose incircle is $o.ur$, illustrated with an outer rectangle in Figure 4. Then, the area($R \cap o.ur$) is getting larger, implying that p_o is also getting larger. Because the incircle's radius is the apothem of the rectangle, the corresponding rectangle becomes $[\mu_{x_1} - r, \mu_{x_1} + r] \times [\mu_{x_2} - r, \mu_{x_2} + r]$ when the uncertain region's circle with the center (μ_{x_1}, μ_{x_2}) and the radius of r.

Now, we are ready to discuss how to compute the value of c_{in} and c_{out} to find corresponding R_{min} and R_{max} under the uniform distribution assumption. We represent the uncertainty region as square where each side's width is $2r$ without loss of generality for simple representation. Under the uniform distribution assumption of uncertainty region which is represented with rectangular shape,

$$p_o = \frac{area(o.ur \cap R)}{area(o.ur)} \quad (6)$$

where *area*() returns the area of the given region.

Finding Minimal c_{in}. Let w and h be the width and height of $o.ur \cap R$, respectively, implying $area(o.ur \cap R) = w \cdot h$. In order to find the minimized value of c_{in}, consider the case where p_o is minimized but still satisfies $p_o \geq p_c$, i.e., for all o_i located within R_{min}, $\min_i(p_{o_i}) \geq p_c$ is satisfied. In this case, we can set $m = \min(w, h)$ without loss of generality. Then, from Equation (6), the inequality $\frac{m^2}{area(o.ur)} \geq p_c$ holds, implying $\frac{(r + c_{in})^2}{\pi r^2} \geq p_c$ since $m = r + c_{in}$ when p_o is minimized. Because we want to find

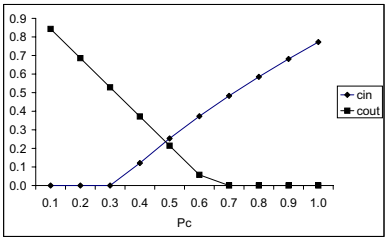


Fig. 5. Effect of p_c when $r = 1$

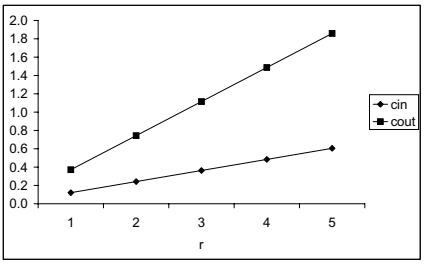


Fig. 6. Effect of r when $p_c = 0.4$

the minimum value of c_{in}, we can set $c_{in} = \max(r(\sqrt{\pi p_c} - 1), 0)$. Therefore, R_{min} is computed by shrinking R by c_{in} in each dimension. One thing to notice is that we cannot fix the value of c_{in} in advance because the size of uncertainty is dependent on the elapsed time after the last update as illustrated in Section 2.1.

Finding Minimal c_{out}. In order to find the minimal value of c_{out}, consider the case where p_o is maximized while still satisfying $p_o \leq p_c$. Let w and h be the width and height of $o.ur \cap R$. Observe that p_o is maximized when $w = 2r$ and $h = r - c_{out}$ or vice versa.[3] In this case, $area(o.ur \cap R) = 2r(r - c_{out})$. Then, the inequality $2r(r - c_{out}) \leq p_c \pi r^2$ holds from Equation (6), which implies $c_{out} \geq (1 - \frac{1}{2}\pi p_c)r$. Because we want to find the minimum value of c_{out}, we can set $c_{out} = \max((1 - \frac{1}{2}\pi p_c)r, 0)$.

Example 1. Given $R = [10, 20] \times [10, 20]$, $p_c = 0.4$, and $r = 1$, c_{in} and c_{out} become $c_{in} = 0.1270$ and $c_{out} = 0.3717$. Thus, R_{min} is $[10.1210, 19.8790] \times [10.1210, 19.8790]$, and $R_{max} = [9.6283, 20.3717] \times [9.6283, 20.3717]$.

Figure 5 and Figure 6 show the experimental results of c_{in} and c_{out} for various values of p_c and r respectively. As Figure 5 illustrates, the value of c_{in} is increased while the value of c_{out} is decreased with respect to the increasing value of p_c when r is fixed as 1. In case of c_{in}, this is expected because in order to have R_{min} guarantee the correctness of the given location predicate evaluation, the size of R_{min} should be smaller (or c_{in} is increased) as p_c is increased. However, the size of R_{max} gets smaller (or c_{out} gets smaller) as the value of p_c is increased, which is also expected because higher threshold value implies that smaller number of objects satisfy this threshold. Figure 6 illustrates that both c_{in} and c_{out} is increased with respect to the increasing value of r when $p_c = 0.4$. This is because both c_{in} and c_{out} has the positive relationship with r in the formulae.

4.3 Evaluation of Imprecise Access Requests

The imprecision of location measures implies that given $\alpha \in \mathcal{P}$, either $se(\alpha)$ if location predicate is included, or $re(\alpha)$, if location predicate is included, cannot be evaluated

[3] This is because of an implicit assumption of the condition $c_{out} \leq r$ since we want to have area(R_{max}) $\leq$ area(R).

deterministically. Instead, it is natural to assign a probability value to the requester's answer to access request results. This approach is similar to uncertain databases [22] where each object in the query results is associated with the probability value such as (o_i, p_i) where o_i is the object and p_i is the quantification of probability that o_i satisfies the query.

Definition 6 (Imprecise Access Request). *An imprecise access request (IAR) is the form of* $\langle user\_id, re, action, p_q \rangle$ *where user_id is the identifier of the user who submits the request, re is a resource expression, action is the requested action, and* p_q *is the probability threshold that the probability quantification* (p_i) *of each resource that evaluates re be true must satisfy (i.e.,* $p_i \geq p_q$ *must holds).*

Given an IAR q, $user\_id(q)$, $re(q)$, $action(q)$, $p_q(q)$ will denote the user, the action, the set of resources evaluated by the resource expression, and p_q of q. The result of IAR is a set of resources that are allowed to gain access to and their quantification probability p_i is greater than or equal to p_q. Given an IAR, LS evaluates $\mathcal{P}$ to find all the relevant rules $\mathcal{P}' \subseteq \mathcal{P}$ that are applicable to the requester. Then, we need to find if $\exists \alpha \in \mathcal{P}'$, the objects specified by re(α) contains each u specified by re(q). Algorithm 1 describes the detailed IAR processing by utilize of the proposed R_{min} and R_{max}.

5 Related Work

Incorporating location information for access control is not a new concept. Atluri and Chun [4] propose an access control model suitable to geo-spatial data. Similarly, Bertino et al. [9] extends the RBAC model to support location-based conditions, called GEO-RBAC, which can deal with mobile data. However, these models do not consider uncertainty within the model, and thus, the access control decision does not guarantee the correctness of the evaluation. There are also several access control models that support protecting people location information in the context of ubiquitous or pervasive computing environment [12, 14]. However, these approaches are different from our approach because they focus on preventing location information from leaking to unauthorized entities by introducing the concept of trust. Actually, Ardagna et al. [3] address the representation and evaluation of location-based access control systems with uncertainty considered, but it does not discuss efficient evaluation of access control requests.

Regarding the uncertainty, Wolfson et al. [23] introduce a cost based approach to determine the size of the uncertainty area. A formal quantitative approach to the aspect of uncertainty in modeling moving objects is presented in [17]. However, the authors limit the uncertainty to the past of the moving objects and the error may become very large as time approaches now. Trajcevski et al. [21] introduced a set of spatiotemporal range queries that apply the uncertainty in traditional range queries. Cheng et al. [10] are the first to formulate the uncertain data retrieval, and contrary to the case of traditional data, uncertain data retrieval involves probabilistic quality with the query results. The work in [11] develops the notion of *x-bounds*, and based on this concept, index-based access methods, called the probability threshold index for one-dimensional uncertain objects. Tao et al. develop a multi-dimensional access method, called the U-Tree [20] which extends [11] to multi-dimensional space.

Algorithm 1. IAR Processing

Input: a set of authorizations $\mathcal{P}$ and IAR q
Output: a set of authorized objects $O' \subseteq O$ where for each $o_i \in O$, $p_i \geq p_q(q)$.
for each $\alpha \in \mathcal{P}$ **do**
  if q is MM or MS **then**
    retrieve the uncertainty region $o.ur$ of $user\_id(q)$ from the LS
    compute R_{max} and R_{min} from $\alpha(R)$
    if loc(o) is contained inR_{min} **then**
      $O_c \leftarrow O_c \cup$ re(α)
    else if loc(o) is contained in $D - R_{max}$ **then**
      do nothing
    else if $p_o \geq p_c$ **then**
      $O_c \leftarrow O_c \cup$ re(α).
    end if
  else
    if se(α) includes $user\_id(q)$ **then**
      $O_c \leftarrow O_c \cup$ re(α)
    end if
  end if
end for
if q is SM or MM **then**
  Compute R_{max} and R_{min} from $\alpha(R_r)$
  Range query of resources located within R_{min}, denoted as O_{c_1}
  Range query of resources location within $R_{max} - R_{min}$, denoted as O_{c_2}.
  $O' \leftarrow$ result of set intersection operation of O_{c_1} and O_c
  $O'' \leftarrow$ result of set intersection operation of O_{c_2} and O_c
  for each $o \in O''$, remove it from O'' if $p_o < p_c$
  return $O' \cup O''$
else
  $O' \leftarrow$ result of set intersection operation of O_c and re(q)
  return O'
end if

6 Conclusions and Future Work

In this paper, we presented a solution to enforce access control policies suitable for a mobile environment with considering location uncertainty. Our proposed solution includes the uncertainty-embedded authorization model, efficient enforcement algorithms, and handling user requests. Our proposed R_{min} and R_{max} can effectively filter out most of the objects from the candidate answer so that evaluation of expensive computation is greatly reduced.

Several open issues still remain. A particularly interesting issue is how to create an index structure for authorizations in order to achieve efficient search process of relevant authorizations given an access request. Most of the currently available authorization enforcement techniques search all the authorization base to find relevant authorizations, which is not efficient obviously. Although there are some work for this direction such as [6, 7, 8, 25], no work has considered uncertainty issue. We would like to develop

enforcement algorithms based on the proposed index structure by utilizing the concepts of R_{min} and R_{max}.

References

[1] Active badge next generation applications, `http://www.cs.agh.edu.pl/ABng/applications.html`

[2] Ardagna, C.A., Cremonini, M., Damiani, E., De Capitani di Vimercati, S., Samarati, P.: Location Privacy Protection Through Obfuscation-based Techniques. In: IFIP TC11/WG 11.3 21st Annual Conference on Data and Applications Security (2007)

[3] Ardagna, C.A., Cremonini, M., Damiani, E., di Vimercati, S.D.C., Samarati, P.: Supporting location-based conditions in access control policies. In: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, pp. 212–222. ACM, New York (2006)

[4] Atluri, V., Chun, S.A.: An Authorization Model for Geospatial Data. IEEE Transactions on Dependable and Secure Computing, 238–254 (2004)

[5] Atluri, V., Chun, S.A.: A geotemporal role-based authorisation system. International Journal of Information and Computer Security 1(1), 143–168 (2007)

[6] Atluri, V., Guo, Q.: Unified Index for Mobile Object Data and Authorizations. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 80–97. Springer, Heidelberg (2005)

[7] Atluri, V., Shin, H.: Efficient Security Policy Enforcement in a Location Based Service Environment. In: Barker, S., Ahn, G.-J. (eds.) Data and Applications Security 2007. LNCS, vol. 4602, pp. 61–76. Springer, Heidelberg (2007)

[8] Atluri, V., Shin, H., Vaidya, J.: Efficient security policy enforcement for the mobile environment. Journal of Computer Security 16(4), 439–475 (2008)

[9] Bertino, E., Catania, B., Damiani, M.L., Perlasca, P.: GEO-RBAC: a spatially aware RBAC. In: Proceedings of the tenth ACM symposium on Access control models and technologies, pp. 29–37. ACM, New York (2005)

[10] Cheng, R., Kalashnikov, D.V., Prabhakar, S.: Evaluating probabilistic queries over imprecise data. In: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pp. 551–562. ACM, New York (2003)

[11] Cheng, R., Xia, Y., Prabhakar, S., Shah, R., Vitter, J.S.: Efficient indexing methods for probabilistic threshold queries over uncertain data. In: Proceedings of the Thirtieth international conference on Very large data bases, vol. 30, pp. 876–887. VLDB Endowment (2004)

[12] Hengartner, U., Steenkiste, P.: Access control to people location information. ACM Transactions on Information and System Security (TISSEC) 8(4), 424–456 (2005)

[13] Horsmanheimo, S., Jormakka, H., Lähteenmäki, J.: Location-Aided Planning in Mobile Network Trial Results. Wireless Personal Communications 30(2), 207–216 (2004)

[14] Kagal, L., Finin, T., Joshi, A.: Trust-Based Security in Pervasive Computing Environments (2001)

[15] Liu, H., Darabi, H., Banerjee, P., Liu, J.: Survey of wireless indoor positioning techniques and systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37(6), 1067–1080 (2007)

[16] Pei, J., Hua, M., Tao, Y., Lin, X.: Query answering techniques on uncertain and probabilistic data: tutorial summary. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 1357–1364. ACM, New York (2008)

[17] Pfoser, D., Jensen, C.S.: Capturing the Uncertainty of Moving-Object Representations. In: Güting, R.H., Papadias, D., Lochovsky, F.H. (eds.) SSD 1999. LNCS, vol. 1651, pp. 111–131. Springer, Heidelberg (1999)

[18] Ray, I., Toahchoodee, M.: A Spatio-temporal Role-Based Access Control Model. In: Barker, S., Ahn, G.-J. (eds.) Data and Applications Security 2007. LNCS, vol. 4602, pp. 211–226. Springer, Heidelberg (2007)

[19] Sistla, P.A., Wolfson, O., Chamberlain, S., Dao, S.: Querying the uncertain position of moving objects. In: Etzion, O., Jajodia, S., Sripada, S. (eds.) Dagstuhl Seminar 1997. LNCS, vol. 1399, p. 310. Springer, Heidelberg (1998)

[20] Tao, Y., Cheng, R., Xiao, X., Ngai, W.K., Kao, B., Prabhakar, S.: Indexing multi-dimensional uncertain data with arbitrary probability density functions. In: Proceedings of the 31st international conference on Very large data bases, pp. 922–933. VLDB Endowment (2005)

[21] Trajcevski, G., Wolfson, O., Zhang, F., Chamberlain, S.: The Geometry of Uncertainty in Moving Objects Databases. In: Jensen, C.S., Jeffery, K., Pokorný, J., Šaltenis, S., Bertino, E., Böhm, K., Jarke, M. (eds.) EDBT 2002. LNCS, vol. 2287, pp. 233–250. Springer, Heidelberg (2002)

[22] Widom, J.: Trio: A system for integrated management of data, accuracy, and lineage. In: CIDR (2005)

[23] Wolfson, O., Yin, H.: Accuracy and Resource Consumption in Tracking and Location Prediction. In: Hadzilacos, T., Manolopoulos, Y., Roddick, J., Theodoridis, Y. (eds.) SSTD 2003. LNCS, vol. 2750, pp. 325–343. Springer, Heidelberg (2003)

[24] Youssef, M., Agrawala, A.: Handling samples correlation in the horus system. In: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, vol. 2 (2004)

[25] Youssef, M., Atluri, V., Adam, N.R.: Preserving mobile customer privacy: an access control system for moving objects and customer profiles. In: Proceedings of the 6th international conference on Mobile data management, pp. 67–76. ACM, New York (2005)

[26] Youssef, M.A., Agrawala, A., Shankar, A.U.: WLAN location determination via clustering and probability distributions. In: IEEE PerCom. 2003, pp. 23–26 (2003)

Using Edit Automata for Rewriting-Based Security Enforcement

Hakima Ould-Slimane[1], Mohamed Mejri[1], and Kamel Adi[2]

[1] Computer Science Department, Laval University, Quebec, (Qc), Canada
[2] Computer Science Department, University of Quebec in Outaouais, (Qc), Canada

Abstract. Execution monitoring (EM) is a widely adopted class of security mechanisms. EM-enforceable security properties are usually characterized by security automata and their derivatives. However Edit automata (EA) have been recently proposed to specify more powerful EMs. Being able to feign the execution of sensitive program actions, these EMs are supposed to enforce more security properties. However, feigning program actions will usually make the program behaving in discordance with its specification since the effects of feigned actions are not reflected in the program states. In this paper we highlight this problem and show how program rewriting[1] can be a reliable enforcement alternative. The paper contribution is mainly a semantics foundation for program rewriting enforcement of EA-enforceable security properties.

Keywords: Execution monitoring, edit automata, security properties, program rewriting.

1 Introduction

Execution Monitoring (EM) is recognized as an efficient and a powerful class of security mechanisms. An EM enforces a security property by intercepting the property-relevant events and performing an intervention procedure when a program is about to violate the property being enforced [2]. However, the efficiency of EM comes with a significant time and memory overhead. The overhead of monitoring programs is generally attributed to the software and/or hardware artifacts needed for (1) intercepting security relevant actions, (2) analyzing the program execution history (trace), and (3) performing the intervention procedure. However a significant overhead can be avoided if the internal structure of the controlled program is known to the EM. Indeed, statically analyzing the controlled program can reveal that many security-relevant invocations, that the EM is controlling by default during execution, are "innocent" and hence can be safely ignored during execution. In addition, the analysis of the execution history can be simplified by taking into consideration the current program step. All these motivated the recent introduction of the program rewriting approach

[1] Program rewriting studied in this paper refers to the class of security enforcement mechanisms introduced in [1] and has no direct relation with rewriting theory.

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 175–190, 2009.

as alternative to EM [3,1]. In this approach, a security property is enforced on a program by rewriting it to produce a more restrictive one that obeys to the enforced property. In [1], program rewriting is identified as an extremely powerful technique which can be used to enforce properties beyond those enforceable by execution monitors.

A commonly adopted rewriting approach consists in taking an EM, usually specified by a security automaton [2], and embedding it into the program [3,4]. Based on the contributions of [2,5], the security enforcement community shared the belief that any EM-enforceable property is a safety, or equivalently a prefix-closed, property. This belief is based on the fact that an EM can intervene to a potential property violation by only halting the execution of the controlled program. However this belief has been disproved later in [6] by recognizing more powerful EMs having the ability of feigning intercepted actions or inserting (executing) actions on behalf of controlled programs. We call them rewriting-based EMs and they are specified using Edit Automata [7,6]. Feigning actions means preventing their execution in a way that cannot be detected by the controlled program. To enforce a non-safety property, an edit automaton feigns the intercepted actions as long as the entire sequence read before does not obey the enforced property. When it recognizes a safe sequence, it reinserts (executes) all the feigned actions in the same order of their interception.

1.1 Problem of Feigned Actions in Security Enforcement

The efficiency of edit automata in security enforcement is due to their ability to feign potentially dangerous actions and actually execute them later when a safe sequence is recognized. However, not all program actions can be feigned. Indeed, this weakness has been clearly mentioned and admitted in [7,6]. In addition, feigning actions should be performed transparently such that (1) it should not be detected by the controlled program and (2) it should maintain the controlled program in a coherent state. Let ϕ_1 be the property stating that any action *inc* incrementing the value of some variable x should be followed immediately by an action sending the value of that variable. Let assume that the EM enforcing ϕ_1 is intercepting the actions of the program P (Version A) below:

```
1: x := 3; y := 4;
2: If (x>1) then inc(y);
3: If (y>4) then send(y);
```

Version A

```
1: x := 3; y := 4;
2: If (x>1) then
3:   If (y+1>4) then
4:              {inc(y); send(y);}
5: Else If (y>4) then   send(y);
```

Version B

When intercepting the $inc(y)$ action (step 2), the EM cannot accept it since it does not know the P structure, so it should feign it. However, since $inc(y)$ has not been executed, the $send(y)$ action (step 3) cannot be performed since $y = 4$ which does not satisfy $y > 4$. In this case the EM prevented a valid program sequence from being executed. This problem cannot be addressed without modifying the program structure. A possible solution is to rewrite the program as

shown in Version B above. In this new version of P, even if the condition of incrementing y is true, i.e., $x > 1$, incrementing y is delayed until the condition of sending y is satisfied, i.e., $y > 4$. However, since the condition $y > 4$ is assumed to be performed on the value of y after incrementation, we modified this condition so it handles $y + 1$ instead of y. This example highlights the limitations of EM enforcement and shows how program rewriting can enforce properties that cannot be enforced by EM.

1.2 Contributions

In this paper we highlight the problem of feigning actions in EM and show how program rewriting can be a reliable enforcement alternative. We propose a rewriting-based formal technique for enforcing security properties on programs. The enforced properties are those targeted by EA. This technique is based on an injection operator over Extended Finite State Machines (EFSMs). Our technique allows to feign actions in a transparent way; a challenging issue which has not been addressed by previous works. Given an untrusted program and an edit automaton enforcing a property, we produce a new version that satisfies the property. In addition, we proved the soundness and the transparency of our technique. Both programs and the edit automata enforcing security properties are specified by EFSMs.

The rest of the paper is structured as follows. Section 2 discusses the related work. Section 3 presents the formal definitions of EFSMs and Edit Automata. Section 4 provides the formal definition of the injection operator. Section 5 gives the main theorems of our approach. Section 6 illustrates the use of our injection operator through an example. Finally, section 7 presents the conclusion and the future work.

2 Related Work

The characterization of EM security enforcement has been first addressed by Schneider in [2]. One important contribution of [2] is the introduction of security automata as formal specification of EM-enforceable security properties. Another contribution of the same work is the identification of prefix-closed properties as the superset of EM-enforceable security properties. The EMs considered by Schneider control the execution of programs, recognize valid executions, and halt the program execution when a security property violation attempt is identified.

In [7,8,9] Ligatti *et al.* investigated enforcing more security properties using EM. To extend the EM enforcement limits identified in [2], they percept EMs as sequence transformers instead of sequence recognizers as adopted by Schneider. The main contribution of this work is the introduction of edit automata to specify more powerful EMs. Ligatti et al. made an assumption that there exists a security-relevant set of actions for which EA can (1) feign the actions execution, and (2) later execute them on behalf of the controlled programs. As consequence, they extend the theoretical limits of EM enforcement from prefix-closed properties to all properties over finite sequences and renewal properties over infinite

sequences [6]. However, practical limitation of EA-based EM is the ability to feign the execution of actions without altering valid executions, an issue that we are addressing by the proposed paper.

Hamlen *et al.* [1] provided a taxonomy of the main security enforcement mechanisms classes. This taxonomy covers static enforcement, execution monitoring, and program rewriting. By connecting the taxonomy to the arithmetic hierarchy of computational complexity theory, the authors succeeded to provide an evaluation of the complexity of each enforcement mechanisms class. The main contribution that is directly related to our work is the identification of inlined EM as alternative to conventional EM. In addition, it showed that program rewriting is an extremely powerful technique which can be used to enforce properties beyond those enforceable by execution monitors [1]. Since the paper was prepared in the same period as [7,6], the authors failed to identify the limitations of EA enforcement compared to program rewriting. Indeed, they (1) recognize EA-enforceable properties as subset of program rewriting properties enforcement (2) but did not add formal constraints on the absence of feigned actions effects on the program execution.

3 Formal Apparatus

In this section we present the main formal definitions needed in this paper. We start by defining extended finite state machines then edit automata.

3.1 Extended Finite State Machine

In order to formally capture the behavior of programs and edit automata, we represent them using Extended Finite State Machines. We adopted this model since the definition of an EFSM is expressive enough to represent edit automata as well as programs with guarded actions that may carry variables.

Definition 1 (Extended Finite State Machine). *Let Σ be a set of possible actions, $\mathcal{V}$ a set of variables, $\mathcal{E}$ a set of arithmetic expressions over $\mathcal{V}$, and $\mathcal{G}$ a set of predicates over Σ^* and $\mathcal{E}$. An Extended Finite State Machine over Σ^*, $\mathcal{V}$, and $\mathcal{G}$ is a 3-tuple (S, i, Δ) where:*

- *S is a finite set of states,*
- *i is a the initial state,*
- *$\Delta \subseteq (S \times (\mathcal{G} \times \Sigma^*) \times S)$ is a transition relation,*

The transition relation Δ consists of 3-tuples of the form $(s_1, (g, \sigma), s_2)$ denoted by $s_1 \xrightarrow{(g,\sigma)} s_2$. This represents a transition from state s_1 to state s_2 guarded by the expression g and executing the sequence of actions σ.

3.2 Edit Automata

Edit automata (EA) are introduced by Ligatti *et al.* [8,6]. In this kind of automata, when given a current state and an input action, the edit automaton

transition function specifies a new state to enter and a sequence of actions to execute. The transition function specifies the intervention action to take in order to enforce the property. The intervention action can be accepting the input action or feigning it, inserting a sequence of actions, or halting the execution.

An EA A enforcing a property ϕ can be specified by an EFSM, denoted by E_ϕ. For each transition $s \xrightarrow{g/\sigma} d$ of E_ϕ (1) the guard g specifies the set of actions that can be intercepted by the transition and a condition on their parameters (2) the sequence σ specifies the intervention action that the EA should undertake as response to an intercepted action. Usually, the guard g has the form $\alpha = Act \mid \alpha \neq Act$ where α is the variable referring to the action intercepted by A and Act is any program action. If α should be accepted by A then $\sigma = \alpha$ meaning that α will be actually executed by E_ϕ. If α should be feigned by A then $\sigma = \varepsilon$ meaning that E_ϕ will do nothing as answer to an attempt to execute α. If some sequence of actions σ' should be inserted by A then $\sigma = \sigma'$ meaning that σ' will be executed by E_ϕ instead of α. The three transitions of the EFSM of Figure 1 are examples of each of the aforementioned cases. The EFSM of Figure 1 specifies an EA A_1 enforcing the property ϕ_1 defined by the following regular expression:

$$L_{\phi_1} = \{((\neg inc(x))^*.inc(x).send(x))^*\}$$

The property ϕ_1 states that any action inc incrementing the value of some variable x should be followed immediately by an action sending the value of x.

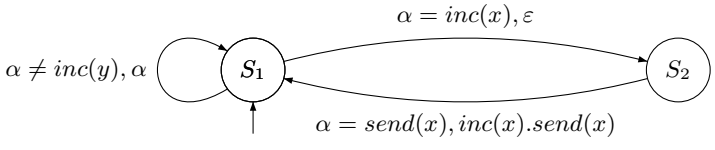


Fig. 1. An EFSM Specifying the EA Enforcing the Property ϕ_1

4 Injection Operator

In this section, we present our injection operator used to enforce a property ϕ on a program $\mathcal{P}$. Intuitively, this operator inject the EA enforcing ϕ in the program $\mathcal{P}$. The injection operator is a kind of synchronous product over automata. However, since it is used for security enforcement, the order of composition is significant.

Definition 2 (Injection). *Let Σ be a set of possible actions, $\mathcal{V}_\mathcal{P}$ and $\mathcal{V}_\phi$ two disjoint sets of variables, and $\mathcal{G}$ a set of boolean propositions over Σ^* and $\mathcal{V}$. Let $E_\mathcal{P} = (S_\mathcal{P}, i_\mathcal{P}, \Delta_\mathcal{P})$ be an EFSM over Σ^*, $\mathcal{V}_\mathcal{P}$ and $\mathcal{G}$ and $E_\phi = (S_\phi, i_\phi, \Delta_\phi)$ an EFSMs over Σ^*, $\mathcal{V}_\phi$ and $\mathcal{G}$. The EFSM $E_\mathcal{P}$ specifies the behavior of a program*

P while the EFSM E_ϕ specifies the EA enforcing a property ϕ. The ***injection modulo functions*** *$\theta, \mathcal{H}, \mathcal{R}$ denoted by $\circlearrowleft^{\theta}_{\mathcal{H},\mathcal{R}}$ of E_ϕ into $E_\mathcal{P}$ produces the EFSM $E_{\phi 2\mathcal{P}} = (S, i, \Delta)$ over Σ^*, $\mathcal{V}$, and $\mathcal{G}$ where:*

- $S \subseteq S_\mathcal{P} \times S_\phi \times \Sigma^*$.
- $i =< i_\mathcal{P}, i_\phi, \varepsilon >$.
- *Δ is the set of transitions such that for any state $(s_i, s_k, h) \in S$, if there exists a transition $(s_i, (g_\mathcal{P}, \sigma_\mathcal{P}), s_j) \in \Delta_\mathcal{P}$ and a transition $(s_k, (g_\phi, \sigma_\phi), s_l) \in \Delta_\phi$ then:*

  - $(< s_i, s_k, h >, (g, \sigma), < s_j, s_l, h' >) \in \Delta$ *if* $\sigma_\mathcal{P} \neq \varepsilon$
  - $(< s_i, s_k, h >, (g, \sigma), < s_j, s_k, h' >) \in \Delta$ *if* $\sigma_\mathcal{P} = \varepsilon$

  where :

  $h' = \mathcal{H}(h, \sigma_\mathcal{P}, \sigma_\phi)$,
  $\mathcal{H} : \Sigma^* \times \Sigma^* \times \Sigma^* \longrightarrow \Sigma^*$,
  and $(g, \sigma) = \mathcal{R}((g_\mathcal{P}, \sigma_\mathcal{P}), (g_\phi, \sigma_\phi), h)$,
  $\mathcal{R} : (\mathcal{G} \times \Sigma^*) \times (\mathcal{G} \times \Sigma^*) \times \Sigma^* \longrightarrow (\mathcal{G} \times \Sigma^*)$,

The effects function θ associates to each action $\alpha \in \Sigma$ its effect $\theta(\alpha)$, which is a substitution representing the execution of α. More formally, let $\mathcal{V}(\alpha)$ be the set of variables affected by the execution of some action α. The effect of the execution of α on $\mathcal{V}(\alpha)$ denoted by $\theta(\alpha) = \{\forall x_i \in \mathcal{V}(\alpha), x_i \backslash E_i)\}$ where E_i denotes an expression over $\mathcal{V}$ that will be assigned as new value to the variable x_i after the execution of α. The followings are examples of actions effects:

$$\theta(inc(x)) = \{x \backslash x + 1\}$$
$$\theta(a(z)) = \{z \backslash z^2 + z + 1\}$$

The function $\mathcal{H}$ is used to associate with each state of $E_{\phi 2P}$ the sequence of actions h feigned so far as required by E_ϕ. The new sequence h' depends on the sequence h feigned so far and the intervention action required by E_ϕ at this step. The function $\mathcal{R}$ is used to compute the guard and the sequence of actions that should be executed by each transition of $E_{\phi 2P}$. The guard and the actions sequence depend on the sequence σ feigned so far by $E_{\phi 2P}$. More details on the definitions of $\mathcal{H}$ and $\mathcal{R}$ will be provided below:

$$\mathcal{H}(h, \sigma_\mathcal{P}, \sigma_\phi) = \begin{cases} h.\sigma_\mathcal{P} & \text{if } \sigma_\phi = \varepsilon \\ \varepsilon & \text{otherwise} \end{cases}$$

The history associated with the initial state i is empty because no action has been feigned so far . For the other states, if $\sigma_\phi = \varepsilon$, which means that the current program action ($\sigma_\mathcal{P}$) should be feigned, then the sequence of actions feigned so far (h) should be extended by $\sigma_\mathcal{P}$. If $\sigma_\phi \neq \varepsilon$ then the intervention action needed

to enforce ϕ is either accepting the current action or inserting some sequence of actions. In both situations, the history of feigned actions should be reset to ε.

The function $\mathcal{R}$ specifies the actual property enforcement and is defined by:

$$\mathcal{R}((g_{\mathcal{P}}, \sigma_{\mathcal{P}}), (g_{\phi}, \sigma_{\phi}), h) = \begin{cases} ((g_{\mathcal{P}} \wedge (\sigma_{\mathcal{P}} \Vdash g_{\phi}))\, \Theta(h), \Omega(\sigma_{\mathcal{P}}, \sigma_{\phi}, h)) & \text{if } \sigma_{\mathcal{P}} \neq \varepsilon \\ ((g_{\mathcal{P}})\, \Theta(h), \varepsilon) & \text{otherwise} \end{cases}$$

where Θ, $\Vdash$, and Ω are defined as follows:

- $\Theta(h)$ is the generalization of function θ on sequences of actions. Indeed, It is the substitution obtained from the history h and applied to the guard $g_{\mathcal{P}}$ and the condition $(\sigma_{\mathcal{P}} \Vdash g_{\phi})$ to generate a more restrictive guard. This new guard carries the impact (the effect) of feigning the actions of h on the program variables. Consequently, we call $\Theta(h)$ "the feigning substitution of h". The use of the substitutions $\Theta(h)$ represents the way we are following to feign actions execution. Following this way, if the actions of h are feigned at some execution point in $E_{\phi 2P}$ then, from the property enforcement (E_{ϕ}) viewpoint, (E_P) will behave according to the EA (E_{ϕ}) recommendations. However from E_P viewpoint, $E_{\phi 2P}$ will be in an incoherent state since it will be in a state were h is supposed to be already executed, which is not the case. To avoid this incoherence, the substitution $\Theta(h)$ is used to simulate the effects of executing h on the program variables. As a result, the actions in $E_{\phi 2P}$ are executed according to the recommendations of E_{ϕ} while the program guards controlling the execution of actions reflect the program state as if the actions are executed exactly as recommended by E_P.
  Let $\Theta(h)$ be the substitution to be applied to some guard g, we have:

  - If $h = \varepsilon$ (the history is empty), then we have: $(g)\,\Theta(\varepsilon) = g$.
  - If $h = \alpha_1.\alpha_2 \ldots \alpha_n$ then: $(g)\,\Theta(\alpha_1.\alpha_2 \ldots \alpha_n) = (g)(\theta(\alpha_1) \circ \ldots \circ \theta(\alpha_n)) = (\ldots(((g)\,\theta(\alpha_n))\theta(\alpha_{n-1}))\ldots\theta(\alpha_1))$. For example:
    $(x > y)\,\Theta(inc(x).dec(y)) = (x > y)\{x \backslash x{+}1\} \circ \{y \backslash y{-}1\} = (x{+}1 > y{-}1)$
    where $inc(x)$ is the action that increments the value of x, while $dec(y)$ is the action that decrements the value of y.

- "$\Vdash$" is called "the Satisfaction Function", it takes a program sequence (consisting of only one action) and a property guard (which concerns some program action) and returns a guard over program variables. The relation "$\Vdash$" is defined in Table 1 where:

  - $Act(g)$ denotes the function extracting the action from a property predicate g. For example: $Act(\alpha = read(z)) = read(z)$.
  - $mgu(a, Act(g))$ denotes the most general unifier between a and $Act(g)$.
  - $\mathcal{V}(a)$ denotes the set of variables used and/or updated by the action a.
  - $\mathcal{V}(g)$ denotes the set of variables involved in the predicate g.

Table 1. The Satisfaction Function

$$a \Vdash g = \begin{cases} tt & \text{if } g = tt \\ \neg(a \Vdash g') & \text{if } g = \neg g' \\ ff & \text{if } mgu(a, Act(g)) \text{ doesn't exist} \\ Tdyn^{\mathcal{V}(g)}_{\mathcal{V}(a)}(\delta) \wedge (Tpar(g))\delta & \text{if } \exists \delta = mgu(a, Act(g)) \end{cases}$$

Table 2. The Dynamic Test

$$\begin{array}{lll} Tdyn^{\mathcal{V}(g)}_{\mathcal{V}(a)}(\emptyset) & = tt & \\ Tdyn^{\mathcal{V}(g)}_{\mathcal{V}(a)}(\{x \mapsto t\} \cup \delta) & = (x = t) \wedge Tdyn^{\mathcal{V}(g)}_{\mathcal{V}(a)}(\delta) & \text{if } x \in \mathcal{V}(a) \wedge t \notin \mathcal{V}(g) \\ Tdyn^{\mathcal{V}(g)}_{\mathcal{V}(a)}(\{x \mapsto y\} \cup \delta) & = Tdyn^{\mathcal{V}(g)}_{\mathcal{V}(a)}(\delta) & \text{if } x \in \mathcal{V}(a) \wedge y \in \mathcal{V}(g) \\ Tdyn^{\mathcal{V}(g)}_{\mathcal{V}(a)}(\{x \mapsto t\} \cup \delta) & = Tdyn^{\mathcal{V}(g)}_{\mathcal{V}(a)}(\delta) & \text{if } x \in \mathcal{V}(g) \end{array}$$

- $Tdyn^{\mathcal{V}(g)}_{\mathcal{V}(a)}(\delta)$ is a boolean condition on $\mathcal{V}(a)$ and it is extracted from the substitution δ. This substitution is obtained from the unification of a program action and the corresponding action targeted by the property guard. This condition represents the dynamic test that should be checked during execution. $Tdyn^{\mathcal{V}(g)}_{\mathcal{V}(a)}(\delta)$ is defined in Table 2. For example:

  $Tdyn^{\{y,z\}}_{\{x\}}([x \mapsto 5, y \mapsto 3, x \mapsto z]) = (x = 5)$.
- $Tpar(g)$ denotes the function extracting the guard handling the property parameters from the predicate g. For example:
  $Tpar(\alpha = read(z) \wedge z > 2) = (z > 2)$. Hence, the condition $(Tpar(g))\delta$ will reflect the test required by the property on the program variables, by substituting the property parameters with the corresponding program variable, as stated by the substitution δ.

The followings are some examples of computing the satisfaction function:

- $a(x) \Vdash (\alpha = a(z)) = \underline{tt}$. This means that the program action $a(x)$ satisfies the property predicate $(\alpha = a(z))$ with no condition.
- $a(x) \Vdash (\alpha = a(3)) = \underline{(x = 3)}$. This means that the program action $a(x)$ satisfies the property predicate $(\alpha = a(3))$ under the condition $(x = 3)$, which is a dynamic test.
- $a(x) \Vdash (\alpha = b(z)) = \underline{ff}$. This means that the program action $a(x)$ does not satisfy the property predicate $(\alpha = b(z))$.

– Ω is the function that generates the new sequence of actions to be executed at each transition of $E_{\phi 2P}$. It is called "the editing function" and defined by:

$$\Omega(\sigma_{\mathcal{P}}, \sigma_{\phi}, h) \quad = \quad \begin{cases} \varepsilon & \text{if } \sigma_{\phi} = \varepsilon \\ \sigma_{\mathcal{P}} & \text{if } \sigma_{\phi} = \text{“}\alpha\text{”} \\ h.\sigma_{\mathcal{P}} & \text{otherwise} \end{cases}$$

The editing function Ω will take one of the three following decisions:

- Feigning the current program action, by executing no action ($\sigma_{\phi} = \varepsilon$)
- Executing the same action as required by the program ($\sigma_{\phi} =$ “α”)
- Executing a sequence of actions when a valid sequence prefix ($h.\sigma_{\mathcal{P}}$) is reached.

The followings are some examples of the use of Ω:

- $\Omega(a(x), \underline{\varepsilon}, b(y).c(z)) = \varepsilon$ (feigning case).
- $\Omega(a(x), \underline{\alpha}, \varepsilon) = a(x)$ (acceptation case).
- $\Omega(b(3), \underline{a(x).b(y)}, a(z)) = a(z).b(3)$ (insertion case).

Optimization. After the construction of the EFSM $E_{\phi 2P}$, some optimizations will be performed on it in order to eliminate unreachable states. This is done by eliminating all the useless transitions. A transition is considered as useless if it is labeled with (g, σ) where g is equivalent to false (ff). If after eliminating useless transitions some nodes become not reachable from the initial state then they will be removed. We remove also each path $(g_m, \sigma_m)...(g_n, \sigma_n)$ in which all the actions have been feigned during the enforcement ($\sigma_m = ... = \sigma_n = \varepsilon$), since the new program will execute no action following those paths.

5 Main Results

The following definitions are needed to prove the theorems presented in this section. Any mention to a path refers to a path starting from the initial state.

Definition 3 (Property Satisfaction). *Let $E_{\mathcal{P}}$ be the EFSM specifying a program $\mathcal{P}$ and E_{ϕ} an EFSM specifying an EA enforcing a property ϕ. We say that $\mathcal{P}$ satisfies Φ, denoted by $\mathcal{P} \vDash \phi$, if every path in E_P can be extended to a path in E_P that satisfies Φ, let $\tau_{\mathcal{P}} = (g_1, \sigma_1).(g_2, \sigma_2), \ldots (g_n, \sigma_n)$ be this path. Since each σ_i is a sequence of actions, we can write their concatenation as: $\sigma_1.\sigma_2 \ldots \sigma_n = a_1.a_2 \ldots a_m$ where each a_i is an atomic action and $m = |\sigma_1.\sigma_2 \ldots \sigma_n|$.*

The path $\tau_{\mathcal{P}}$ satisfies Φ, if there exists a path $\tau_{\phi} = (g'_1, \sigma'_1).(g'_2, \sigma'_2) \ldots (g'_m, \sigma'_m)$ belonging to E_{Φ} such that:

(A) $mgu(a_1 \ldots a_m, \sigma'_1 \ldots \sigma'_m) \neq \emptyset$.

(B) $g_1 \sqsubseteq (a_1 \Vdash g'_1)$ *and* $\forall 1 \leq j \leq m. \exists 1 \leq i \leq n$ *such that* $a_1 \ldots . a_j$ *is a prefix of* $\sigma_1 \ldots . \sigma_i$ *and* $(g_1 \bigwedge_{2 \leq l \leq i}(g_l) \Theta(\sigma_1 \ldots \sigma_{l-1})) \sqsubseteq (a_j \Vdash g'_j) \Theta(a_1 \ldots a_{j-1})$

where $\sqsubseteq$ is an ordering relation over boolean expressions. For any two conditions b and b', we say that $b \sqsubseteq b'$, meaning “b is more restrictive than b'”, if there exists a condition b'', such that $b = b' \wedge b''$.

Intuitively, condition (A) states that the execution produced by the program path matches a prefix of a property path regardless of the property conditions. Condition (B) states that the conjunction of all the conditions (guards) controlling the execution of any action of the program at some prefix of the path is more restrictive than the conjunction of those conditions (guards) expected by the property. Since the number of the conditions of the program path can be different from the number of conditions of the property path, the effects of the actions executed before some program/property condition are simulated on them.

Definition 4 (Precise Program Restriction $\precsim$). *Let $E_{\mathcal{P}}$ and $E_{\mathcal{P}'}$ be two EFSMs specifying two programs $\mathcal{P}$ and $\mathcal{P}'$ respectively. We say that $\mathcal{P}$ is a precisely more restrictive version of $\mathcal{P}'$, denoted by $\mathcal{P} \precsim \mathcal{P}'$, if each path belonging to $E_{\mathcal{P}}$ can be extended to some path $\tau_{\mathcal{P}} = (g_1, \sigma_1) \dots (g_n, \sigma_n)$ belonging to $E_{\mathcal{P}}$ for which there exists a path $\tau_{\mathcal{P}'} = (g'_1, \sigma'_1) \dots (g'_m, \sigma'_m)$ belonging to $E_{\mathcal{P}'}$ such that:*

(C) $(g_1 \wedge \bigwedge_{2 \leq i \leq n}(g_i)\Theta(\sigma_1 \dots \sigma_{i-1})) \sqsubseteq (g'_1 \wedge \bigwedge_{2 \leq i \leq m}(g'_i)\Theta(\sigma'_1 \dots \sigma'_{i-1}))$[2]

(D) $\sigma_1 \dots \sigma_n = \sigma'_1 \dots \sigma'_m$.

Intuitively, condition (C) states that the conjunction of all the conditions (guards) controlling the execution produced by the path of program $\mathcal{P}$ is more restrictive than the conjunction of those conditions controlling the same execution produced by the path of program $\mathcal{P}'$. Condition (D) states that any execution of $\mathcal{P}$ can be performed by $\mathcal{P}'$. The definition of "precise program restriction" is needed by Theorem 1 to compare a program with the program resulting from injecting a property into it. Here the resulting program should be more restrictive than the original one in terms of the possible executions and the conditions controlling them. Definition 4 is also needed by Theorem 2 to verify that a any subprogram of $\mathcal{P}$ that satisfies a property is preserved by the property injection.

Definition 5 (Conservative Program Restriction $\preceq$). *Let $E_{\mathcal{P}}$ and $E_{\mathcal{P}'}$ be two EFSMs specifying two programs $\mathcal{P}$ and $\mathcal{P}'$ respectively. We say that $\mathcal{P}$ is a conservatively more restrictive version of $\mathcal{P}'$, denoted by $\mathcal{P} \preceq \mathcal{P}'$, if for each path $\tau_{\mathcal{P}} = (g_1, \sigma_1) \dots (g_n, \sigma_n)$ belonging to $E_{\mathcal{P}}$ there exists some path $\tau_{\mathcal{P}'} = (g'_1, \sigma'_1) \dots (g'_n, \sigma'_n)$ belonging to $E_{\mathcal{P}'}$ such that:*

(E) $g_1 \sqsubseteq g'_1 \wedge \forall 2 \leq i \leq n.(g_i)\,\Theta(\sigma_1 \dots \sigma_{i-1}) \sqsubseteq (g'_i)\,\Theta(\sigma'_1 \dots \sigma'_{i-1})$.

(F) $\forall i, 1 \leq i \leq n. \quad \sigma_i = \sigma'_i$.

Intuitively, conditions (E) and (F) together state that for any path of $\mathcal{P}$ there exists some path in $\mathcal{P}'$ (1) executing the same sequence of actions, (2) having the same number of guards that each of which controls the same sequence of

[2] Recall that Θ is the substitution simulating the effect of a sequence of actions on the transitions predicates. See Section 4 for the formal definition of Θ.

action, and (3) each guard of $\mathcal{P}$ is more restrictive than the corresponding guard of $\mathcal{P}'$. The definition of "conservative program restriction" is needed by Theorem 2 to identify inside a program $\mathcal{P}$ all its subprograms that satisfy a property. The theorem will select these subprograms as conservative restrictive versions of $\mathcal{P}$.

The following theorem states that our approach of injecting a property ϕ into a program $\mathcal{P}$ is sound. Soundness means that all the possible executions of the EFSM resulting from injecting ϕ into $\mathcal{P}$ are possible executions of the original EFSM specifying $\mathcal{P}$. It states also that all the possible executions of the resulting EFSM satisfies the property ϕ.

Theorem 1 (Soundness). *Let $\mathcal{P}$ be a program specified by the EFSM $E_{\mathcal{P}}$, Φ a property enforced by the EA specified by the EFSM E_ϕ, and $P \circlearrowleft \Phi$ be the program specified by the EFSM $E_{\phi 2\mathcal{P}} = E_{\mathcal{P}} \circlearrowleft^{\theta}_{\mathcal{H},\mathcal{R}} E_\phi$. We have:*

(I) $P \circlearrowleft \Phi \precsim P$

(II) $P \circlearrowleft \Phi \vDash \Phi$

Proof. [3]
Let $\tau = (g_1, \sigma_1).(g_2, \sigma_2) \dots (g_n, \sigma_n)$ be any path of $E_{\phi 2\mathcal{P}}$. It follows, from the definition of a path, that there exist states $(s_0, h_0), (s_1, h_1) \dots (s_{n-1}, h_{n-1})$, (s_n, h_n) such that $(s_0, h_0) \xrightarrow{(g_1,\sigma_1)} (s_1, h_1) \dots (s_{n-1}, h_{n-1}) \xrightarrow{(g_n,\sigma_n)} (s_n, h_n)$ is a legitimate sequence of transitions, starting from (s_0, h_0). From the definition of $\circlearrowleft^{\theta}_{\mathcal{H},\mathcal{R}}$, it follows that there exist states $s_0^{\mathcal{P}}, s_1^{\mathcal{P}}, \dots s_n^{\mathcal{P}}$ and $s_0^{\phi}, s_1^{\phi}, \dots s_n^{\phi}$ and propositions $g_1^{\mathcal{P}}, g_2^{\mathcal{P}}, \dots g_n^{\mathcal{P}}$ and $g_1^{\phi}, g_2^{\phi}, \dots g_n^{\phi}$ such that:

a) there exists a path $\tau_{\mathcal{P}} = (g_1^{\mathcal{P}}, \sigma_1^{\mathcal{P}}).(g_2^{\mathcal{P}}, \sigma_2^{\mathcal{P}}) \dots (g_n^{\mathcal{P}}, \sigma_n^{\mathcal{P}})$ generated by the following legitimate sequence of transitions, starting from the initial state $s_0^{\mathcal{P}}$ of $E_{\mathcal{P}}$: $s_0^{\mathcal{P}} \xrightarrow{(g_1^{\mathcal{P}},\sigma_1^{\mathcal{P}})} s_1^{\mathcal{P}} \xrightarrow{(g_2^{\mathcal{P}},\sigma_2^{\mathcal{P}})} s_2^{\mathcal{P}} \dots \xrightarrow{(g_n^{\mathcal{P}},\sigma_n^{\mathcal{P}})} s_n^{\mathcal{P}}$.

b) there exists a path $\tau_\phi = (g_1^{\phi}, \sigma_1^{\phi}).(g_2^{\phi}, \sigma_2^{\phi}) \dots (g_n^{\phi}, \sigma_n^{\phi})$ generated by the following legitimate sequence of transitions, starting from the initial state s_0^{ϕ} of E_ϕ: $s_0^{\phi} \xrightarrow{(g_1^{\phi},\sigma_1^{\phi})} s_1^{\phi} \xrightarrow{(g_2^{\phi},\sigma_2^{\phi})} s_2^{\phi} \dots \xrightarrow{(g_n^{\phi},\sigma_n^{\phi})} s_n^{\phi}$

c) from the definition of $\circlearrowleft^{\theta}_{\mathcal{H},\mathcal{R}}$ we have:

- $\forall i \geq 1$: $g_i = (g_i^{\mathcal{P}} \wedge (\sigma_i^{\mathcal{P}} \Vdash g_i^{\phi}))\Theta(h_{i-1})$
- $\forall i \geq 1$: $\sigma_i = \Omega(\sigma_i^{\mathcal{P}}, \sigma_i^{\phi}, h_{i-1})$

Proof of (I): To prove (I): $P \circlearrowleft \Phi \precsim P$ we will prove that for each path τ of $E_{\phi 2\mathcal{P}}$, there exists an extension for which there exists a path $\tau_{\mathcal{P}}$ of $E_{\mathcal{P}}$ such that the conditions (C) and (D) of Definition 4 should be satisfied. Let τ be the extension of τ that will be considered by our proof[4]. From (a), we know that for each path τ there exists some path $\tau_{\mathcal{P}}$ of $E_{\mathcal{P}}$. Accordingly the conditions (C) and (D) to be proved are respectively the following:

[3] For space limitations not all the intermediate steps of the proof have been presented.
[4] Recall that any path can be considered as extension of itself.

– $(g_1 \wedge \bigwedge_{2\leq i\leq n}(g_i)\Theta(\sigma_1 \ldots \sigma_{i-1})) \sqsubseteq (g_1^{\mathcal{P}} \wedge \bigwedge_{2\leq i\leq m}(g_i^{\mathcal{P}})\Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}}))$
– $\sigma_1 . . . . . \sigma_n = \sigma_1^{\mathcal{P}} . . . . . \sigma_n^{\mathcal{P}}$

By replacing the terms by there definitions in c) the two conditions (C) and (D) will be the followings:

(C) $(g_1^{\mathcal{P}} \wedge (\sigma_1^{\mathcal{P}} \Vdash g_1^{\phi})) \wedge \bigwedge_{2\leq i\leq n}(g_i^{\mathcal{P}} \wedge (\sigma_i^{\mathcal{P}} \Vdash g_i^{\phi}))\, \Theta(h_{i-1})$
$\Theta(\Omega(\sigma_1^{\mathcal{P}},\sigma_1^{\phi}, h_0) \ldots \Omega(\sigma_{i-1}^{\mathcal{P}},\sigma_{i-1}^{\phi}, h_{i-2})) \sqsubseteq g_1^{\mathcal{P}} \wedge \bigwedge_{2\leq i\leq n}(g_i^{\mathcal{P}})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}})$.
(D) $\sigma_1 . . . . . \sigma_n = \sigma_1^{\mathcal{P}} . . . . . \sigma_n^{\mathcal{P}}$

d) From the definition of $\mathcal{H}$, we have:
$g_1 = (g_1^{\mathcal{P}} \wedge (\sigma_1^{\mathcal{P}} \Vdash g_1^{\phi}))\, \Theta(h_0) = (g_1^{\mathcal{P}} \wedge (\sigma_1^{\mathcal{P}} \Vdash g_1^{\phi}))\, \Theta(\varepsilon) = g_1^{\mathcal{P}} \wedge (\sigma_1^{\mathcal{P}} \Vdash g_1^{\phi}) \sqsubseteq g_1^{\mathcal{P}}$

Let E_i denote the individual terms at the left side of condition (C):
$(g_i^{\mathcal{P}} \wedge (\sigma_i^{\mathcal{P}} \Vdash g_i^{\phi}))\, \Theta(h_{i-1})\, \Theta(\Omega(\sigma_1^{\mathcal{P}}, \sigma_1^{\phi}, h_0) \ldots \Omega(\sigma_{i-1}^{\mathcal{P}}, \sigma_{i-1}^{\phi}, h_{i-2}))$.

According to the value of the history h_{i-1}, we distinguish two cases:

– *Case 1:* $h_{i-1} = \varepsilon$ (i.e., no action has been feigned so far), we have: $\Omega(\sigma_1^{\mathcal{P}}, \sigma_1^{\phi}, h_0)) \ldots \Omega(\sigma_{i-1}^{\mathcal{P}}, \sigma_{i-1}^{\phi}, h_{i-2}) = \sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}}$. Then we deduce that: $E_i = (g_i^{\mathcal{P}})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}}) \wedge (\sigma_i^{\mathcal{P}} \Vdash g_i^{\phi})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}}) \sqsubseteq (g_i^{\mathcal{P}})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}})$.
– *Case 2:* $h_{i-1} \neq \varepsilon$, i.e., a sequence of actions has been feigned without being reinserted. This sequence is $h_{i-1} = \sigma_m^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}}$ where $1 \leq m \leq i-1$. Consequently, we have: $\Omega(\sigma_1^{\mathcal{P}}, \sigma_1^{\phi}, h_0) \ldots \Omega(\sigma_{i-1}^{\mathcal{P}}, \sigma_{i-1}^{\phi}, h_{i-2}) = \sigma_1^{\mathcal{P}} \ldots \sigma_{m-1}^{\mathcal{P}}$. Then, we deduce that the expression E_i becomes:
$(g_i^{\mathcal{P}})\, \Theta(\sigma_m^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{m-1}^{\mathcal{P}}) \wedge (\sigma_i^{\mathcal{P}} \Vdash g_i^{\phi})\, \Theta(\sigma_m^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{m-1}^{\mathcal{P}}) = (g_i^{\mathcal{P}})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}}) \wedge (\sigma_i^{\mathcal{P}} \Vdash g_i^{\phi})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}}) \sqsubseteq (g_i^{\mathcal{P}})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}})$.

From *Case 1* and *Case 2*, we obtain:
$\forall 2 \leq i \leq n.\, \bigwedge_{2\leq i\leq n} E_i \sqsubseteq \bigwedge_{2\leq i\leq n}(g_i^{\mathcal{P}})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}})$.

By adding the term that we got from d) we obtain:
$(g_1^{\mathcal{P}} \wedge (\sigma_1^{\mathcal{P}} \Vdash g_1^{\phi})) \wedge \bigwedge_{2\leq i\leq n}(g_i^{\mathcal{P}} \wedge (\sigma_i^{\mathcal{P}} \Vdash g_i^{\phi}))\, \Theta(h_{i-1})\, \Theta(\Omega(\sigma_1^{\mathcal{P}}, \sigma_1^{\phi}, h_0) \ldots$
$\Omega(\sigma_{i-1}^{\mathcal{P}}, \sigma_{i-1}^{\phi}, h_{i-2})) \sqsubseteq g_1^{\mathcal{P}} \wedge \bigwedge_{2\leq i\leq n}(g_i^{\mathcal{P}})\, \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{i-1}^{\mathcal{P}})$ which satisfies condition (C).

Since $h_n \neq \varepsilon$, we have: $\Omega(\sigma_1^{\mathcal{P}}, \sigma_1^{\phi}, h_0) \ldots \Omega(\sigma_n^{\mathcal{P}}, \sigma_n^{\phi}, h_{n-1}) = \sigma_1^{\mathcal{P}} \ldots \sigma_n^{\mathcal{P}}$ which satisfies condition (D).

By proving (C) and (D) we have proved that (I): $P \circlearrowleft \Phi \precsim P$.

Proof of (II): To prove (II): $P \circlearrowleft \Phi \vDash \Phi$ we will prove that for each path $\tau = (g_1, \sigma_1).(g_2, \sigma_2) \ldots (g_n, \sigma_n)$ of $E_{\phi 2\mathcal{P}}$, there exists an extension for which there exists a path τ_ϕ of E_ϕ such that the conditions (A) and (B) of Definition 3 should be satisfied. Let τ be the extension of τ that will be considered

by our proof[5]. From a), we know that for each path τ there exists some path $\tau_{\mathcal{P}} = (g_1^{\mathcal{P}}, \sigma_1^{\mathcal{P}}).(g_2^{\mathcal{P}}, \sigma_2^{\mathcal{P}}) \ldots (g_n^{\mathcal{P}}, \sigma_n^{\mathcal{P}})$. such that $\sigma_1 \ldots . sigma_n = \sigma_1^{\mathcal{P}} \ldots . \sigma_n^{\mathcal{P}}$. In addition, from b) we know that for each path τ there exists some path $\tau_\phi = (g_1^\phi, \sigma_1^\phi).(g_2^\phi, \sigma_2^\phi) \ldots (g_n^\phi, \sigma_n^\phi)$. Consequently, the conditions (A) and (B) to be proved are respectively the following:

(A) $mgu(\sigma_1^{\mathcal{P}} \ldots \sigma_n^{\mathcal{P}}, \sigma_1^\phi \ldots \sigma_n^\phi) \neq \emptyset$.
(B) $g_1 \sqsubseteq (\sigma_1^{\mathcal{P}} \Vdash g_1^\phi)$ and $\forall 1 \leq j \leq n . \exists 1 \leq i \leq n$ such that $\sigma_1^{\mathcal{P}} \ldots . \sigma_j^{\mathcal{P}}$ is a prefix of $\sigma_1 \ldots . \sigma_i$ and $g_1 \bigwedge_{2 \leq l \leq i} (g_l) \Theta(\sigma_1 \ldots \sigma_{l-1})) \sqsubseteq (\sigma_j^{\mathcal{P}} \Vdash g_j^\phi) \Theta(\sigma_1^{\mathcal{P}} \ldots \sigma_{j-1}^{\mathcal{P}})$

To prove (A), according to c), we have to prove that For all $1 \leq i \leq n$ there exists $i \leq j \leq n$, such that:
$mgu(\Omega(\sigma_1^{\mathcal{P}}, \sigma_1^\phi, h_0) \ldots \Omega(\sigma_n^{\mathcal{P}}, \sigma_n^\phi, h_{n-1}), \sigma_1^\phi \ldots \sigma_n^\phi) \neq \emptyset$.

e) Since $h_n = \varepsilon$, we have: $\Omega(\sigma_1^{\mathcal{P}}, \sigma_1^\phi, h_0)) \ldots \Omega(\sigma_n^{\mathcal{P}}, \sigma_n^\phi, h_{n-1}) = \sigma_1^{\mathcal{P}} \ldots \sigma_n^{\mathcal{P}}$,
f) From the definitions of functions $\Vdash$ and Ω there exists a substitution Δ, such that: $\sigma_1^{\mathcal{P}} \ldots \sigma_n^{\mathcal{P}} = (\sigma_1^\phi \ldots \sigma_n^\phi)\, \Delta$, where $\sigma_n^\phi \neq \varepsilon$ (accepting or inserting case).
g) From e) and f), we get:
$mgu(\Omega(\sigma_1^{\mathcal{P}}, \sigma_1^\phi, h_0)) \ldots \Omega(\sigma_i^{\mathcal{P}}, \sigma_i^\phi, h_{i-1}), (\sigma_1^\phi \ldots \sigma_i^\phi)) \neq \emptyset$ which satisfies (A).

The proof of (B) is based on term reductions that are similar to those used in the proof of condition (C) in (I). For space limitation, this proof will not presented in the paper.

Hereafter, we formulate the transparency theorem stating that any possible execution of the original EFSM, that respects the security property, is a possible execution of the EFSM resulting from injection.

Theorem 2 (Transparency). [6] *Let P and $\mathcal{P}'$ be two programs specified by the EFSMs $E_{\mathcal{P}}$ and $E_{\mathcal{P}'}$ respectively. Let Φ a property enforced by the EA specified by the EFSM E_ϕ and $P \circlearrowleft \Phi$ be the program specified by the EFSM $E_{\phi 2 \mathcal{P}} = E_{\mathcal{P}} \circlearrowleft^{\theta}_{\mathcal{H},\mathcal{R}} E_\phi$. We have:*

$$\textit{If } \mathcal{P}' \preceq \mathcal{P} \textit{ and } \mathcal{P}' \vDash \Phi \textit{ then } \mathcal{P}' \precsim \mathcal{P} \circlearrowleft \Phi$$

6 Example

In this section, we show through an example how we can use our approach to generate a program satisfying a property from one that did not satisfy it. In this example, we take an EFSM E_{ϕ_2} specifying the EA enforcing a property ϕ_2 (Figure 2) and an EFSM $E_{\mathcal{P}_1}$ representing some program $\mathcal{P}_1$ (Figure 3) not satisfying ϕ_2. Indeed, the sequences $a(x)c(x)$ that can be executed following the path $T_1 \xrightarrow{(x>1, a(x))} T_2 \xrightarrow{(x>=5, \varepsilon)} T_3 \xrightarrow{(y>0, c(x))} T_4$ are examples of traces that are not satisfying ϕ_2 and can result from executing $\mathcal{P}_1$. The effects of the four actions a(x),b(x),c(x), and d(x), involved in $E_{\mathcal{P}_1}$ and E_{ϕ_2} are the followings:

[5] Recall that any path can be considered as extension of itself.
[6] For space limitation, the proof of this theorem has not been presented.

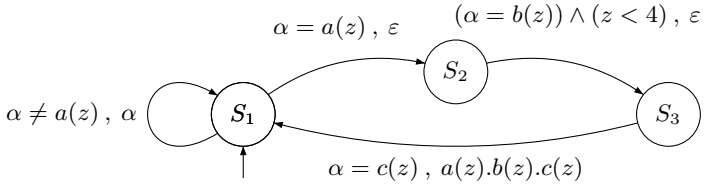


Fig. 2. The EFSM Specifying the EA Enforcing the Property ϕ_2

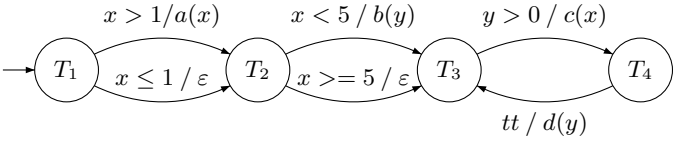


Fig. 3. The EFSM $\mathrm{E}_{\mathcal{P}_1}$

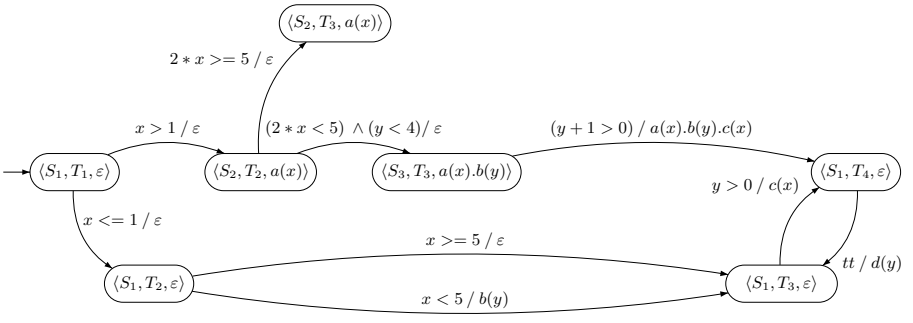


Fig. 4. The EFSM Resulting From Injecting E_{ϕ_2} into E_{P_1}

- $\theta(a(x)) = \{x/2 * x\}$,
- $\theta(b(x)) = \{x/x + 1\}$,
- $\theta(c(x)) = \{x/x^3\}$,
- $\theta(d(x)) = \{x/x - 1\}$.

The injection of E_{ϕ_2} into E_{P_1} produces the EFSM depicted in Figure 4. After simplifying transitions by evaluating transitions guards, and by eliminating (1) transitions guarded by conditions that can be reduced to false, and (2) every path in which all the transitions actions are equal to ε, we got the optimized EFSM. Notice that in this new version of the resulting EFSM, the guards have been modified as follows:

- In the guard $2 * x < 5$, the action $a(x)$ has been feigned,
- In the guard $y + 1 > 0$, the action $b(y)$ has been feigned,

– The guard $y < 4$ is a dynamic test that guaranties that the argument of the action $b(y)$ will satisfy the condition required by the property ϕ_2.

7 Conclusion

In this paper, we present a rewriting-based formal approach for enforcing security properties. These properties are supposed to be enforceable by those EM that are specifiable by edit automata [9]. Being able to feign the execution of sensitive actions of controlled programs, edit automata are supposed to enforce a wide range of properties that cannot be enforced by conventional EM. However, feigning program actions will usually make the program behaving in discordance with its specification. The reason of this discordance is the fact that the effects of feigned actions are not reflected in the program states. In this paper, we highlighted this problem and showed how program rewriting can be a reliable enforcement alternative. The proposed formal framework used in our approach is based on EFSMs. EFSMs are expressive enough to specify edit automata as well as programs. We defined an injection operator that takes an EFSM specifying an edit automaton and embeds it into an EFSM representing a given program. The EFSM obtained from the injection is a secure version of the original program. We provided the formal results related to our operator as well as their corresponding proofs.

We are currently investigating the use of aspect oriented programming (AOP) [10] to implement EA-based program rewriting enforcement. The theoretical foundations of this ongoing task are based on the pointcut-advice model while its implementation is based on AspectJ [11]. One interesting thread of this work is building a catalogue of real-world security properties and the corresponding aspects enforcing them. Another future research thread related to the paper contribution is leveraging the existing state of the art of conformance testing for EA-based program rewriting enforcement. This thread is motivated by the fact that existing techniques are not adequate to deal with feigning actions and emulating their effects on rewritten programs. The expected techniques will generate the data required to test the conformance of an enforcement mechanism implementation to its formal specification.

References

1. Hamlen, K.W., Morrisett, G., Schneider, F.B.: Computability classes for enforcement mechanisms. ACM Trans. Program. Lang. Syst. 28(1), 175–205 (2006)
2. Schneider, F.B.: Enforceable security policies. ACM Transactions on Information and Systems Security 3(1), 30–50 (2000)
3. Erlingsson, U., Schneider, F.B.: Sasi enforcement of security policies: a retrospective. In: Proceedings of the New Security Paradigms Workshop, Caledon Hills, Ontario, Canada, pp. 87–95. ACM Press, New York (2000)
4. Evans, D., Twyman, A.: Flexible Policy-Directed Code Safety. In: IEEE Symposium on Security and Privacy, Oakland, California (May 1999)

5. Viswanathan, M.: Foundations for the Run-time Analysis of Software Systems. PhD thesis, University of Pennsylvania (2000)
6. Ligatti, J., Bauer, L., Walker, D.: Enforcing non-safety security policies with program monitors. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 355–373. Springer, Heidelberg (2005)
7. Ligatti, J., Bauer, L., Walker, D.: Edit automata: Enforcement mechanisms for run-time security policies. International Journal of Information Security 4(1-2), 2–16 (2005) (published online 26 October 2004)
8. Bauer, L., Ligatti, J., Walker, D.: More enforceable security policies. In: Foundations of Computer Security, Copenhagen, Denmark, July 25-26, pp. 95–104 (2002)
9. Ligatti, J., Bauer, L., Walker, D.: Enforcing non-safety security policies with program monitors. Technical Report TR-720-05, Princeton University (January 2005)
10. Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J.: Aspect-oriented programming. In: Akşit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
11. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.: An overview of aspectJ. In: Knudsen, J.L. (ed.) ECOOP 2001. LNCS, vol. 2072, p. 327. Springer, Heidelberg (2001)
12. Lamport, L.: Proving the correctness of multiprocess programs. IEEE Transactions of Software Engineering 3(2), 125–143 (1977)
13. Hamlen, K., Morrisett, G., Schneider, F.: Computability classes for enforcement mechanisms. Technical Report TR2003-1908, Cornell University (2003); To appear in ACM Transactions on Programming Languages and Systems
14. Necula, G.C.: Proof-carrying code. In: Conference Record of POPL 1997: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, January 1997, pp. 106–119 (1997)

Distributed Anonymization: Achieving Privacy for Both Data Subjects and Data Providers

Pawel Jurczyk and Li Xiong

Emory University, Atlanta GA 30322, USA

Abstract. There is an increasing need for sharing data repositories containing personal information across multiple distributed and private databases. However, such data sharing is subject to constraints imposed by privacy of individuals or data subjects as well as data confidentiality of institutions or data providers. Concretely, given a query spanning multiple databases, query results should not contain individually identifiable information. In addition, institutions should not reveal their databases to each other apart from the query results. In this paper, we develop a set of decentralized protocols that enable data sharing for horizontally partitioned databases given these constraints. Our approach includes a new notion, *l-site-diversity*, for data anonymization to ensure anonymity of data providers in addition to that of data subjects, and a distributed anonymization protocol that allows independent data providers to build a virtual anonymized database while maintaining both privacy constraints.

1 Introduction

Current information technology enables many organizations to collect, store, and use various types of information about individuals in large repositories. Government and organizations increasingly recognize the critical value and opportunities in sharing such a wealth of information across multiple distributed databases.

Problem scenario. An example scenario is the Shared Pathology Informatics Network (SPIN)[1] initiative by the National Cancer Institute. The objective is to establish an Internet-based *virtual* database that will allow investigators access to data that describe archived tissue specimens across multiple institutions while still allowing those institutions to maintain local control of the data. There are some important privacy considerations in such a scenario. First, personal health information is protected under the Health Insurance Portability and Accountability Act (HIPAA)[2,3] and cannot be revealed without de-identification

[1] Shared Pathology Informatics Network. http://www.cancerdiagnosis.nci.nih.gov/spin/

[2] Health Insurance Portability and Accountability Act (HIPAA). http://www.hhs.gov/ocr/hipaa/

[3] State law or institutional policy may differ from the HIPAA standard and should be considered as well.

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 191–207, 2009.

or anonymization. Second, institutions cannot reveal their private databases to each other due to confidentiality of the data. In addition, the institutions may not want to reveal the ownership of their records even if the records are anonymized.

These scenarios can be generalized into the problem of privacy-preserving data publishing for multiple distributed databases where multiple data custodians or providers wish to publish an integrated view of the data for querying purposes while preserving privacy for both *data subjects* and *data providers*. We consider two privacy constraints in the problem. The first is the privacy of individuals or *data subjects* (such as the patients) which requires that the published view of the data should not contain individually identifiable information. The second is the privacy of *data providers* (such as the institutions) which requires that data providers should not reveal their private data or the ownership of the data to each other besides the published view.

Existing and potential solutions. Privacy preserving data publishing or data anonymization for a single database has been extensively studied in recent years. A large body of work contributes to algorithms that transform a dataset to meet a privacy principle such as k-anonymity using techniques such as generalization, suppression (removal), permutation and swapping of certain data values so that it does not contain individually identifiable information [1].

There are a number of potential approaches one may apply to enable data anonymization for distributed databases. A naive approach is for each data provider to perform data anonymization independently as shown in Fig. 1a. Data recipients or clients can then query the individual anonymized databases or an integrated view of them. One main drawback of this approach is that data is anonymized before the integration and hence will cause the data utility to suffer. In addition, individual databases reveal their ownership of the anonymized data.

An alternative approach assumes an existence of third party that can be trusted by each of the data owners as shown in Fig. 1b. In this scenario, data owners send their data to the trusted third party where data integration and anonymization are performed. Clients then can query the centralized database. However, finding such a trusted third party is not always feasible. Compromise of the server by hackers could lead to a complete privacy loss for all the participating parties and data subjects.

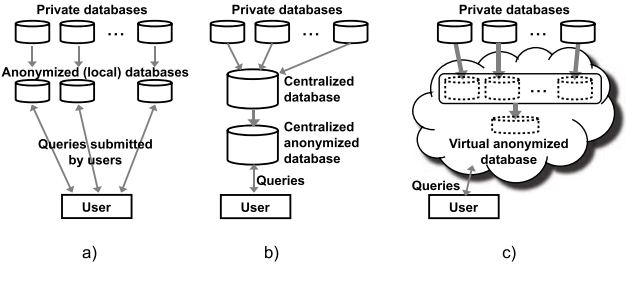


Fig. 1. Architectures for privacy preserving data publishing

In this paper, we propose a distributed data anonymization approach as illustrated in Fig. 1c. In this approach, data providers participate in distributed protocols to produce a *virtual* integrated and anonymized database. Important to note is that the anonymized data still resides at individual databases and the integration and anonymization of the data is performed through the secure distributed protocols. The local anonymized datasets can be unioned using secure union protocols [2,3] and then published or serve as a virtual database that can be queried. In the latter case, each individual database can execute the query on its local anonymized dataset, and then engage in distributed secure union protocols to assemble the results that are guaranteed to be anonymous.

Contributions. We study the problem of data anonymization for horizontally partitioned databases in this paper and present the distributed anonymization approach for the problem. Our approach consists of two main contributions.

First, we propose a *distributed anonymization protocol* that allows multiple data providers with horizontally partitioned databases to build a virtual anonymized database based on the integration (or union) of the data. As the output of the protocol, each database produces a local anonymized dataset and their union forms a virtual database that is guaranteed to be anonymous based on an anonymization principle. The protocol utilizes *secure multi-party computation* protocols for sub-operations such that information disclosure between individual databases is minimal during the virtual database construction.

Second, we propose a new notion, *l-site-diversity*, to ensure anonymity of data providers in addition to that of data subjects for anonymized data. We present heuristics and adapt existing anonymization algorithms for $l - site - diversity$ so that anonymized data achieve better utility.

Organization. The remainder of this paper is organized as follows. Section 2 briefly reviews work related to our research. Section 3 discusses the privacy model we are using and presents our new notion on $l - site - diversity$. Section 4 presents our distributed anonymization protocol. Section 5 presents a set of experimental evaluations and Section 6 concludes the paper.

2 Related Work

Our work is inspired and informed by a number of areas. We briefly review the closely related areas below and discuss how our work leverages and advances the current state-of-the-art techniques.

Privacy preserving data publishing. Privacy preserving data publishing for centralized databases has been studied extensively [1]. One thread of work aims at devising privacy principles, such as k-anonymity, l-diversity, t-closeness, and m-invariance, that serve as criteria for judging whether a published dataset provides sufficient privacy protection. Another large body of work contributes to algorithms that transform a dataset to meet one of the above privacy principles (dominantly k-anonymity). In this study, our distributed anonymization protocol

is built on top of the k-anonymity and l-diversity principles and the greedy top-down Mondrian multidimensional k-anonymization algorithm [4].

There are some works focused on data anonymization of distributed databases. [5] presented a two-party framework along with an application that generates k-anonymous data from two vertically partitioned sources without disclosing data from one site to the other. [6] proposed provably private solutions for k-anonymization in the distributed scenario by maintaining end-to-end privacy from the original customer data to the final k-anonymous results.

In contrast to the above work, our work is aimed at horizontal data distribution and arbitrary number of sites. More importantly, our anonymization protocol aims to achieve anonymity for both data subjects and data providers.

Secure multi-party computation. Our approach also has its roots in the secure multi-party computation (SMC) problem [7,8,9,10,11]. This problem deals with a setting where a set of parties with private inputs wish to jointly compute some function of their inputs. An SMC protocol is *secure* if no participant learns anything more than the output.

Our problem can be viewed as designing SMC protocols for anonymization that builds virtual anonymized database and query processing that assembles query results. Our distributed anonymization approach utilizes existing secure SMC protocols for subroutines such as computing sum [12], the kth element [13], and set union [2,3]. The protocol is carefully designed so that the intermediate information disclosure is minimal.

3 Privacy Model

In this section we present the privacy goals that we focus on in this paper, followed by models and metrics for characterizing how these goals are achieved, and propose a new notion for protecting anonymity for data providers. As we identified in Section 1, we have two privacy goals. First, the privacy of individuals or data subjects needs be protected, i.e. the published virtual database and query results should not contain individually identifiable information. Second, the privacy of data providers needs to be protected, i.e. individual databases should not reveal their data or their ownership of the data apart from the virtual anonymized database.

Privacy for data subjects based on anonymity. Among the many privacy principles that protect against individual identifiability, the seminal works on k-anonymity [14,15] require that a set of k records (entities) to be indistinguishable from each other based on a quasi-identifier set. Given a relational table T, attributes are characterized into: *unique identifiers* which identify individuals; *quasi-identifier (QID)* which is a minimal set of attributes (X_1, ..., X_d) that can be joined with external information to re-identify individual records; and *sensitive attributes* that should be protected. The set of all tuples containing identical values for the QID set is referred to as an *equivalence class*. An improved principle,

l-diversity [16], demands every group to contain at least l well-represented sensitive values.

Given our research goals of extending the anonymization techniques and integrating them with secure computation techniques to preserve privacy for both data subjects and data providers, we based our work on k-anonymity and l-diversity to achieve anonymity for data subjects. While we realize they are relatively weak compared to principles such as differential privacy, the reason we chose them for this paper is that they are intuitive and have been justified to be useful in many practical applications such as privacy-preserving location services. Therefore, techniques enforcing them in the distributed environment will still be practically important. In addition, their fundamental concepts serve as a basis for many other principles and there is a rich set of algorithms for achieving k-anonymity and l-diversity. We can study the subtle differences and effects of different algorithms and their interactions with secure multi-party computation protocols. Finally, our protocol structure, and the underlying concepts will be orthogonal to these privacy principles, and our framework will be extensible so as to easily incorporate more advanced privacy principles.

Privacy for data providers based on secure multi-party computation Our second privacy goal is to protect privacy for data providers. It resembles the goal of secure multi-party computation (SMC). In SMC, a protocol is *secure* if no participant can learn anything more than the result of the function (or what can be derived from the result). It is important to note that for practical purposes, we may relax the security goal for a tradeoff for efficiency. Instead of attempting to guarantee absolute security in which individual databases reveal nothing about their data apart from the virtual anonymized database, we wish to minimize data exposure and achieve a sufficient level of security.

We also adopt the *semi-honest* adversary model commonly used in SMC problems. A semi-honest party follows the rules of the protocol, but it can attempt to learn additional information about other nodes by analyzing the data received during the execution of the protocol. The *semi-honest* model is realistic for our problem scenario where multiple organizations are collaborating with each other to share data and will follow the agreed protocol to get the correct result for their mutual benefit.

Privacy for data providers based on anonymity: a new notion. Now we will show that a method of simply coupling the above anonymization principles and the secure multi-party computation principles is insufficient in our scenario. While the secure multi-party computation can be used for the anonymization to preserve privacy for data providers during the anonymization, the anonymized data itself (considered as results of the secure computation) may compromise the privacy of data providers. The data partitioning at distributed data sources and certain background knowledge can introduce possible attacks that may reveal the ownership of some data by certain data providers. We illustrate such an attack, a homogeneity attack, through a simple example.

Table 1. Illustration of Homogeneity Attack for Data Providers

ID	City	Age	Disease
1	New York	30-40	Heart attack
2	New York	30-40	AIDS

Node 0

ID	City	Age	Disease
3	Northeast	40-43	AIDS
4	Northeast	40-43	Flu

Node 1

Table 1 shows anonymized data that satisfies 2-anonymity and 2-diversity at two distributed data providers (QID: City, Age; sensitive attribute: Disease). Even if SMC protocols are used to answer queries, given some background knowledge on data partitioning, the ownership of records may be revealed. For instance, if it is known that records from *New York* are provided only by *node 0*, then records with ID 1 and 2 can be linked to that node directly. In consequence, privacy of data providers is compromised. Essentially, the compromise is due to the anonymized data and cannot be solved by secure multi-party computation. One way to fix the problem is to generalize the location for records 1 and 2 so that they cannot be directly linked to a particular data provider.

To address such a problem, we propose a new notion, *l-site-diversity*, to enhance privacy protection for data providers. We define a *quasi-identifier set* with respect to data providers as a minimal set of attributes that can be used with external information to identify the ownership of certain records. For example, the location is a QID with respect to data providers in the above scenario as it can be used to identify the ownership of the records based on the knowledge that certain providers are responsible for patients from certain locations. The parameter, l, specifies minimal number of distinct sites that records in each equivalence class belong to. This notion protects the anonymity of data providers in that each record can be linked to at least l providers. Formally, the table T^* satisfies *l-site-diversity* if for every equivalence class g in T^* the following condition holds:

$$count(distinct\ nodes(g)) \geq l \tag{1}$$

where *nodes(g)* returns node IDs for every record in group g.

It can be noted that our definition of *l-site-diversity* is closely related to *l-diversity*. The two notions, however, have some subtle differences. l-diversity protects a data subject from being linked to a particular sensitive attribute. We can map $l-site-diversity$ to l-diversity if we treat the data provider that owns a record as a sensitive attribute for a record. However, in addition to protecting the ownership for a data record as in l-diversity, l-site-diveristy also protects the *anonymity* of the ownership for data providers. In other words, it protects a data provider from being linked to a particular data subject. The QID set with respect to data providers for l-site-diversity could be completely different from the QID set with respect to data subjects for k-anonymity and l-diversity. *l-site-diversity* is only relevant when there are multiple data sources and it adds another check when data is being anonymized so that the resulting data will not reveal the ownership of the records. It is worth mentioning that we could also exploit much stronger definitions of l-diversity such as *entropy l-diversity* or *recursive (c,l)-diversity* as defined in [16].

4 Distributed Anonymization Protocol

In this section we describe our distributed anonymization approach. We first describe the general protocol structure and then present the distributed anonymization protocol.

We assume that the data are partitioned horizontally among n sites ($n > 2$) and each site owns a private database d_i. The union of all the local databases, denoted d, gives a complete view of all data ($d = \bigcup d_i$). In addition, the *quasi-identifier* of each local database is uniform among all the sites. The sites engage in a distributed anonymization protocol where each site produces a local anonymized dataset a_i and their union forms a virtual database that is guaranteed to be k-anonymous. Note that a_i is not required to be k-anonymous by itself. When users query the virtual database, each individual database executes the query on a_i and then engage in a distributed querying protocol to assemble the results that are guaranteed to be k-anonymous.

4.1 Selection of Anonymization Algorithm

Given our privacy models, we need to carefully adapt or design new anonymization algorithms with additional check for site-diversity and implement the algorithm using multi-party distributed protocols. Given a centralized version of anonymization algorithm, we can decompose it and utilize SMC protocols for sub-routines which are provably secure in order to build a secure distributed anonymization protocol. However, performing one secure computation, and using those results to perform another, may reveal intermediate information that is not part of the final results even if each step is secure. Therefore, an important consideration for designing such protocols is to minimize the disclosure of intermediate information.

There are a large number of algorithms proposed to achieve k-anonymity. These k-anonymity algorithms can be also easily extended to support *l-diversity* check [16]. However, given our design goal above, not all anonymization algorithms are equally suitable for a secure multi-party computation. Considering the two main strategies, top-down partitioning and bottom-up generalization, we discovered that top-down partitioning approaches have significant advantages over bottom-up generalization ones in a secure multi-party computation setting because anything revealed during the protocol as intermediate results will in fact have a coarser view than the final result and can be derived from the final result not violating the security requirement.

Based on the rationale above, our distributed anonymization protocol is based on the multi-dimensional top-down Mondrian algorithm [4]. The Mondrian algorithm uses a greedy top-down approach to recursively partition the (multidimensional) quasi-identifer domain space. It recursively chooses the split attribute with the largest normalized range of values, and (for continuous or ordinal attributes) partitions the data around the median value of the split attribute. This process is repeated until no allowable split remains, meaning that the data points in a particular region cannot be divided without violating the anonymity constraint, or constraints imposed by value generalization hierarchies.

Algorithm 1. Distributed anonymization algorithm - leading site ($i = 0$)

1: **function** split(set d_0, ranges of QID attributes)
2: **Phase 1**: Determine split attribute and split point
3: Select best split attribute a (see text)
4: If split is possible, send *split attribute* to node 1. Otherwise, send *finish splitting* to node 1 and finish.
5: Compute median of chosen a for splitting (using secure k-th element algorithm).
6: **Phase 2**: Split current dataset
7: Send a and m to node 1
8: Split set d_0, create two sets, s_0 containing items *smaller than* m and g_0 containing items *greater than* m. Distribute median items among s_i and g_i.
9: Send *finished* to node 1
10: Wait for *finished* from last node (synchronization)
11: **Phase 3**: Recursively split sub datasets
12: Find $size_{left} = |\bigcup s_i|$ and $size_{right} = |\bigcup g_i|$ (using *secure sum* protocol)
13: If further split of left (right) subgroup is possible, send split_left=true (split_right=true) to node 1 and call the split function recursively (updating ranges of QID attributes). Otherwise send split_left=false (split_right=false) to node 1.
14: **end function** split

4.2 Distributed Anonymization Protocol

The key idea for the distributed anonymization protocol is to use a set of secure multi-party computation protocols to realize the Mondrian method for the distributed setting so that each database produces a local anonymized dataset which may not be k-anonymous itself, but their union forms a virtual database that is guaranteed to be k-anonymous. We present the main protocol first, followed by important heuristics that is used in the protocol.

We assume a leading site is selected for the protocol. The protocols for the leading and other sites are presented in Algorithms 1 and 2. The steps performed at the leading site are similar to the centralized Mondrian method. Before the computation starts, range of values for each quasi-identifier in set $d = \bigcup d_i$ and the total number of data points need to be calculated. A *secure kth element* protocol can be used to securely compute the minimum (k=1) and maximum ($k = n$ where n is the total number of tuples in the current partition) values of each attribute across the databases [13].

In Phase 1, the leading site selects the best split attribute and determines the split point for splitting the current partition. In order to select the best split attribute, the leading site uses a heuristic rule that is described in details below. If required, all the potential split attributes (e.g., the attributes that produce subgroups satisfying $l-site-diversity$) are evaluated and the best one is chosen. In order to determine the split medians, a *secure kth element* protocol is used ($k = \lceil \frac{n}{2} \rceil$) with respect to the data across the databases. To test whether given attribute can be used for splitting, we calculate a number of distinct sites in subgroups that would result from splitting on this attribute using the secure sum algorithm. The split is considered possible if records in both subgroups are

Algorithm 2. Distributed anonymization algorithm - non-leading node ($i > 0$)

1: **function** split(set c)
2: Read split attribute a and median m from node $(i-1)$; pass them to node $(i+1)$
3: **if** finish splitting received **then return**
4: Split set c into s_i containing items *smaller than* m and g_i containing items *greater than* m. Distribute median items among s_i and g_i.
5: Read finished from node $i-1$ (sychronization); Send finished to node $i+1$
6: Read $split\_left$ from node $i-1$ and pass it to node $i+1$
7: **if** split_left **then call** split(s_i)
8: Read $split\_right$ from node $i-1$, Send $split\_right$ to node $i+1$
9: **if** split_right **then call** split(g_i)
10: **end function** split

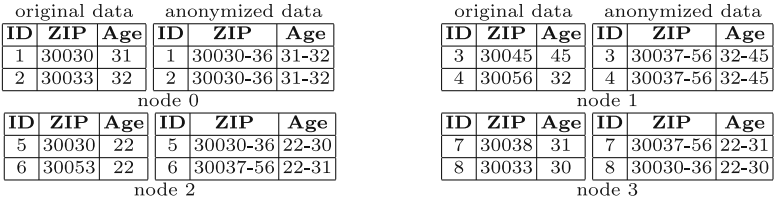


Fig. 2. Distributed anonymization illustration

provided by at least l sites. In Phase 2, the algorithm performs split and waits for all the nodes to finish splitting. Finally in Phase 3, the node recursively checks whether further split of the new subsets is possible. In order to determine whether a partition can be further split, a *secure sum* protocol [12] is used to compute the number of tuples of the partition across the databases.

We illustrate the overall protocol with an example scenario shown in Figure 2 where we have 4 nodes and we use $k = 2$ for k-anonymization and $l = 1$ for *l-site-diversity*. Note that the anonymized databases at node 2 and node 3 are not 2-anonymous by themselves. However the union of all the anonymized databases is guaranteed to be 2-anonymous.

Selection of split attribute. One key issue in the above protocol is the selection of split attribute. The goal is to split the data as much as possible while satisfying the privacy constraints so as to maximize discernibility or utility of anonymized data. The basic Mondrian method uses the range of an attribute as a goodness indicator. Intuitively, the larger the spread, the easier the good split point can be found and more likely the data can be further split. In our setting, we also need to take into account the site diversity requirement and adapt the selection heuristic. The importance of doing so is demonstrated in Figure 3. Let's assume that we want to achieve 2-anonymity and 2-site-diversity. In the first scenario, the attribute for splitting is chosen only based on range of the QID attributes. The protocol finishes with 2 groups of 5 and 4 records (further split is impossible due to 2-site-diversity requirement). The second scenario

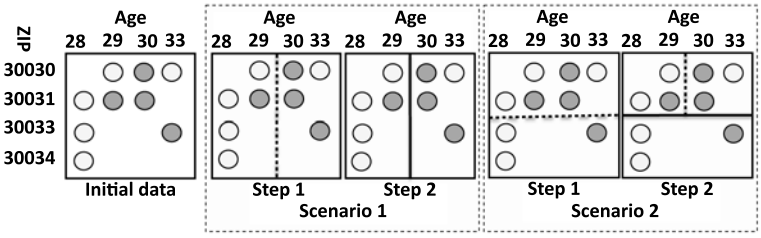


Fig. 3. Impact of split attribute selection when *l-site-diversity* ($n = 2$) is considered. Different shades represent different owners of records.

exploits information on records distribution when the decision on split attribute is made (the more evenly the records are distributed across sites in resulting subgroups, the better). This rule yields better results, namely three groups of 3 records each.

Based on the illustration, intuition suggests that we need to select a split attribute that results in partitions with even distribution of records from different data providers. This makes further splits more likely while meeting the l-site-diversity constraint. Similar to decision tree classifier construction [17], information gain can be used as a scoring metric for selecting attribute that results in partitions with most diverse distribution of data providers. Note that this is used in a complete opposite sense from decision tree where the goal is to partition the data into partitions with homogeneous classes. The information gain of a potential splitting attribute a_k is computed through the information entropy of resulting partitions:

$$e(a_k) = -\sum_{i=0}^{n-1} p_{(i,l_k)} log(p_{(i,l_k)}) - \sum_{i=0}^{n-1} p_{(i,r_k)} log(p_{(i,r_k)}) \tag{2}$$

where l_k and r_k are partitions created after splitting the input set using attribute a_k (and its median value) and $p(i, g)$ is the portion of records that belong to node i in group g. It is important to note that the calculations need to take into account data on distributed sites and thus secure sum protocol needs to be used.

Our final scoring metric combines the original range value based metric and the new diversity-aware metrics using a linear combination as follows:

$$\forall_{a_i \in Q} s_i = \alpha \frac{range(a_i)}{\max\limits_{a_j \in Q}(range(a_j))} + (1 - \alpha) \frac{e(a_i)}{\max\limits_{a_j \in Q}(e(a_j))} \tag{3}$$

where *range* function returns range of attribute, $e(a_i)$ returns values of information entropy as defined above when attribute a_i is used for splitting and α is a weighting parameter.

Important to note is that if *l-site-diversity* is not required (e.g., *l=1*), then the evaluation of the heuristic rule above is limited to checking only the range of attributes, and choosing the attribute with the widest range.

4.3 Analysis

Having presented the distributed anonymization protocol, we analyze the protocol in terms of its security and overhead.

Security. We will now analyze the security of our distributed k-anonymity protocol. Our proofs will show that, given the result, the leaked information (if any), and the site's own input, any site can simulate the protocol and everything that was seen during the execution. Since the simulation generates everything seen during execution of the protocol, clearly no one learns anything new from the protocol when it is executed. The proofs will also use a general composition theorem [7] that covers algorithms implemented by running many invocation of secure computations of simpler functionalities. Let's assume a *hybrid model* where the protocol uses a trusted third-party to compute the result of such smaller functionalities $f_1 ... f_n$. The composition theorem states that if a protocol in hybrid model is secure in terms of comparing the real computation to the ideal model, then if a protocol is changed in such a way that calls to trusted third-party are replaced with secure protocols, the resulting protocol is still secure.

We have analyzed the distributed k-anonymity protocol in terms of security and present the following two theorems with sketches of proofs. For complete proofs, we refer readers to [18].

Theorem 1. The distributed k-anonymity protocol privately computes a k-anonymous view of horizontally partitioned data in semi-honest model when $l = 1$.

Proof sketch. The proof needs to show that any given node can simulate the algorithm and all what was seen during its execution given only the final result and its local data. The simulation first analyzes the complete anonymized data and finds a range of each attribute from the quasi identifier. This can be done by looking for the largest and the smallest possible values of all the QID attributes. Next, as no $l-site-diversity$ is required (l=1), the site knows that the attribute used for splitting is actually the attribute with the largest range. As the site knows which attribute was used for splitting, it can attempt to identify the split point using the following approach. First, it identifies all possible splitting attributes/points that could be used when the algorithm was executed. Formally, the points of potential split are the distinct values that appear on the bounds of the QID ranges in the anonymized view. To identify the median value (or the value that was used for split) the node checks which of the potential splitting points is actually a median. This can be done by choosing a value that divides the set of records into two sets with the sizes closest to half of the number of records in the database (note that the subsets resulting from spitting might not have equal sizes - for instance if number of records is odd). Now the site is ready to simulate the split. If size of any of the two groups that result from splitting is greater than or equal to $2 * k$, this group can be further split. In such case the node would continue the described simulation with input data being one of the subgroups.

Theorem 2. The distributed k-anonymity protocol privately computes a k-anonymous view of horizontally partitioned data in semi-honest model, revealing at most the following statistics of the data, when $l > 1$.

1. Median values of each attribute from QID for groups of records of size $\geq 2*k$,
2. Entropy of distribution of records for groups resulting from potential splits,
3. Number of distinct sites that provide data to groups resulting from potential splits (the identity of those sites are confidential).

Proof sketch. The proof uses a similar approach as above. The main difference is in the decision step when a node uses the entropy based heuristic designed for l-site-diversity to decide on the split attribute. In this case, not only the range of attribute, but also the distribution of records in groups resulting from potential splitting, need to be considered. Using the final result, information from points 1, 2 and 3, and the range of QID attributes, however, any node can decide on the split attribute in the protocol simulation.

Overhead. Our protocol introduces additional overhead due to the fact that the nodes have to use additional protocols in each step of computation. The time complexity of the original Mondrian algorithm is $O(nlogn)$ where n is the number of items in the anonymized dataset [4]. As we presented in Algorithm 1, each iteration of the distributed anonymization algorithm requires calculation of the heuristic decision rule, median value of an attribute, and the count of tuples of a partition. The secure sum protocol does not depend on the number of tuples in the database. The secure $k - th$ element algorithm is logarithmic in number of input items (assuming the worst - case scenario that all the input items are distinct). As a consequence, the time complexity of our protocol can be estimated as $O(nlog^2 n)$ in terms of number of records in a database.

The communication overhead of the protocol is determined by two factors. The first is the cost for a single round. This depends on the number of nodes involved in the system and the topology which is used and in our case it is proportional to the number of nodes on the ring. As the future work, we are considering alternative topologies (such as trees) in order to optimize the communication cost for each round. The second factor is the number of rounds and is determined by the number of iterations and the sub-protocols used by each iteration of the anonymization protocol. The secure sum protocol involves one round of communication. In the secure kth element protocol, the number of rounds is $logM$ (M being the range of attribute values) and each round requires secure computations twice. It is important to note that the distributed anonymization protocol is expected to be run offline on an infrequent basis. As a result, the overhead of the protocol will not be a major issue.

5 Experimental Evaluation

We have implemented the distributed anonymization protocol in Java within the DObjects framework [19] which provides a platform for querying data across

distributed and heterogeneous data sources. To be able to test a large variety of configurations, we also implemented the distributed anonymization protocol using a simulation environment. In this section we present a set of experimental evaluations of the proposed protocols.

The questions we attempt to answer are: 1) What is the advantage of using distributed anonymization algorithm over centralized or independent anonymization? 2) What is the impact of the *l-site-diversity* constraint on anonymization protocol? 3) What are the optimal values for α parameter in our heuristic rules presented in equation 3?

5.1 Distributed Anonymization vs. Centralized and Independent Anonymization

We first present an evaluation of the distributed anonymization protocol compared to the centralized and independent anonymization approaches in terms of the quality of the anonymized data.

Dataset and setup. We used the Adult dataset from UC Irvine Machine Learning Repository. The dataset contained 30161 records and was configured as in [4]. We used 3 distributed nodes (30161 records were split among those nodes using round-robin protocol). We report results for the following scenarios: 1) the data is located in one centralized database and classical Mondrian k-anonymity algorithm was run (centralized approach), 2) data are distributed among the three nodes and Mondrian k-anonymity algorithm was run at each site independently (independent or naive approach) and 3) data are distributed among the three nodes and we use the distributed anonymization approach presented in section 4. We ran each experiment for different k values. All the experiments in this subsection used *1-site-diversity*.

Results. Figure 4 shows the average equivalence class size with respect to different values of k. We observe that our distributed anonymization protocol performs the same as the non-distributed version. Also as expected, the naive approach (independent anonymization of each local database) suffers in data utility because the anonymization is performed before the integration of the data.

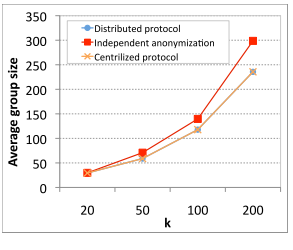


Fig. 4. Average equivalence class size vs. k

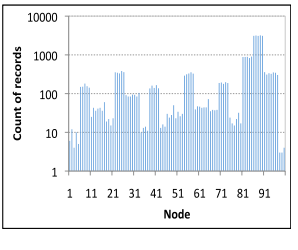


Fig. 5. Histogram for partitioning using City and Age

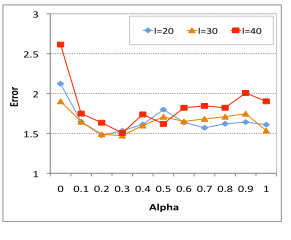


Fig. 6. Average error vs. α (information gain-based)

5.2 Achieving Anonymity for Data Providers

The experiments in this section again use the Adult dataset. The data is distributed across $n = 100$ sites unless otherwise specified. We experimented with distribution pattern that we will describe in detail below.

Metric. The average equivalence group size as shown in previous subsection provides a general data utility metric. The query imprecision metric provides an application-specific metric that is of particular relevance to our problem setting. Given a query, since the attribute values are generalized, it is possible only to return the tuples from the anonymized dataset that are contained in any generalized ranges overlapping with the selection predicate. This will often produce a larger result set than evaluating the predicate over the original table. For this set of experiments, we use summary queries (queries that return count of records) and we use an algorithm similar to the approach introduced in [20] that returns more accurate results. We report a *relative error* of the query results. Specifically, given *act* as an exact answer to query and *est* as an answer computed according to algorithm defined above, the relative error is defined as $|act - est|/act$. For each of the tested configurations, we submit 10,000 randomly generated queries, and for each query we calculate a relative error. We report average value of the error. Each query uses predicates on two randomly chosen attributes from *quasi-identifier*. For boolean attributes that can have only two values (e.g. sex), the predicate has a form of $a_i = value$. For other attributes we use predicate in the form $a_i \in R$. R is a random range and has a length of $0.3 * |a_i|$, where $|a_i|$ denotes the domain size of an attribute.

Data partitioning. In a realistic scenario, data is often split according to some attributes. For instance, the patient data can be split according to cities, i.e. the majority records from a hospital located in New York would have a New York address while those from a hospital located in Boston would have a Boston address. Therefore, we distributed records across sites using partitioning based on attribute values. The rules of partitioning were specified using two attributes, City and Age. The dataset contained data from 6 different cities, and every 1/6th of available nodes were assigned to a different city. Next, records within each group of nodes for a given city were distributed using Age attribute: records with age less than 25 were assigned to the first 1/3rd of nodes, records with age between 25 and 55 to the second 1/3rd of nodes, and the remaining records to the remaining nodes. The histogram of the records per node in this setup is presented in Figure 5 (please note the logarithmic scale of the plot).

Results. We now present the results evaluating the impact of α value under this setup. Figure 6 presents the average query error for different α and l values for the heuristic rule we used. We can observe a significant impact of α value on the average error. The smallest error value is observed for $\alpha = 0.3$ and this seems to be an optimal choice for all tested l values. One can observe 30% decrease in error when compared to using only range as in original Mondrian ($\alpha = 1.0$) or using only diversity-aware metrics ($\alpha = 0.0$). It is worth mentioning that we

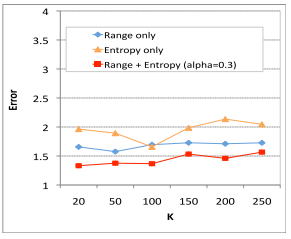


Fig. 7. Average error vs. k ($l = 30$)

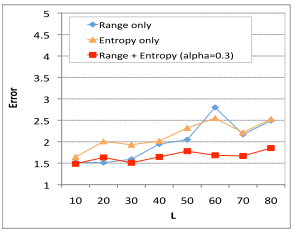


Fig. 8. Average error vs. l ($k = 200$)

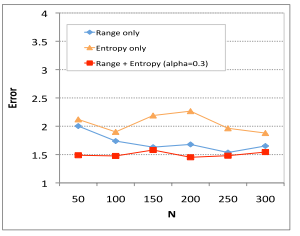


Fig. 9. Error vs. n ($k = 200$ and $l = 30$)

have also experimented with different distributions of records, and the results were consistent with what we presented above. We do not provide these results due to space limitations.

The next experiment was focused on the impact of k parameter on average error. We present results for *l=30* in Figure 7 for three different split heuristic rules: using range only, information gain only, and combining range with information gain with $\alpha = 0.3$. We observe that the heuristic rule that takes into account both range and information gain gives consistently the best results and a reduction of error around 30%. These results do not depend on the value of k.

Next, we tested the impact of the l parameter for $l - site - diversity$. Figure 8 shows an average error for varying l and $k = 200$ using the same heuristic rules as in previous experiment. Similarly, the rule that takes into account range and information gain gives the best results. With increasing l, we observe an increasing error rate because the data needs to be more generalized in order to satisfy the diversity constraints.

So far we have tested only scenarios with 100 nodes ($n = 100$). To complete the picture, we plot the average error for varying n ($k = 200$ and $l = 30$) in Figure 9. One can notice that the previous trends are maintained - the results do not appear to be dependent on the number of nodes in the system. Similary, the rule that takes into account range and information gain is superior to other methods and the query error is on average 30% smaller than that for others.

6 Conclusion

We have presented a distributed and decentralized anonymization approach for privacy-preserving data publishing for horizontally partitioned databases. Our work addresses two important issues, namely, privacy of data subjects and privacy of data providers. We presented a new notion, *l-site-diversity*, to achieve anonymity for data providers in anonymized dataset. Our work continues along several directions. First, we are interested in developing a protocol toolkit incorporating more privacy principles and anonymization algorithms. In particular, dynamic or serial releases of data with data updates are extremely relevant in our distributed data integration setting. Such concepts as *m-invariance* [20] or

l-scarsity [21] are promising ideas and we plan to extend our research in this direction. Second, we are also interested in developing specialized multi-party protocols such as set union that offer a tradeoff between efficiency and privacy as compared to the existing set union protocols based on cryptographic approaches.

Acknowledgement

We thank Kristen LeFevre for providing us the implementation for the Mondrian algorithm and the anonymous reviewers for their valuable feedback. The research is partially supported by a URC and an ITSC grant from Emory and a Career Enhancement Fellowship from the Woodrow Wilson Foundation.

References

1. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey on recent developments. ACM Computing Surveys (in press)
2. Kantarcioglu, M., Clifton, C.: Privacy preserving data mining of association rules on horizontally partitioned data. IEEE Transactions on Knowledge and Data Engineering (TKDE) 16(9) (2004)
3. Böttcher, S., Obermeier, S.: Secure set union and bag union computation for guaranteeing anonymity of distrustful participants. JSW 3(1), 9–17 (2008)
4. LeFevre, K., DeWitt, D., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: Proceedings of the International Conference on Data Engineering (ICDE 2006) (2006)
5. Jiang, W., Clifton, C.: A secure distributed framework for achieving k-anonymity. VLDB Journal 15(4), 316–333 (2006)
6. Zhong, S., Yang, Z., Wright, R.N.: Privacy-enhancing k-anonymization of customer data. In: Proc. of the Principles of Database Systems (PODS) (2005)
7. Goldreich, O.: Secure multi-party computation, Working Draft, Version 1.3 (2001)
8. Clifton, C., Kantarcioglu, M., Vaidya, J.: Tools for privacy preserving distributed data mining. ACM SIGKDD Explorations 4 (2003)
9. Lindell, Y., Pinkas, B.: Secure multiparty computation for privacy-preserving data mining. Cryptology ePrint Archive, Report 2008/197 (2008), `http://eprint.iacr.org/`
10. Vaidya, J., Clifton, C.: Privacy-preserving data mining: Why, how, and when. IEEE Security & Privacy 2(6), 19–27 (2004)
11. Du, W., Atallah, M.J.: Secure multi-party computation problems and their applications: a review and open problems. In: NSPW 2001: Proceedings of the 2001 workshop on New security paradigms, pp. 13–22. ACM, New York (2001)
12. Schneier, B.: Applied Cryptography, 2nd edn. John Wiley & Sons, Chichester (1996)
13. Aggarwal, G., Mishra, N., Pinkas, B.: Secure computation of the kth-ranked element. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 40–55. Springer, Heidelberg (2004)
14. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. 13(6), 1010–1027 (2001)
15. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 10(5), 557–570 (2002)

16. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. In: Proceedings of the International Conference on Data Engineering (ICDE 2006), p. 24 (2006)
17. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2006)
18. Jurczyk, P., Xiong, L.: Distributed anonymization: Achieving privacy for both data subjects and data providers. Technical Report TR-2009-013, Emory University Department of Mathematics and Computer Science (2009)
19. Jurczyk, P., Xiong, L.: Dobjects: Enabling distributed data services for metacomputing platforms. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008, Part I. LNCS, vol. 5101, pp. 136–145. Springer, Heidelberg (2008)
20. Xiao, X., Tao, Y.: M-invariance: towards privacy preserving re-publication of dynamic datasets. In: Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 689–700 (2007)
21. Bu, Y., Fu, A.W.C., Wong, R.C.W., Chen, L., Li, J.: Privacy preserving serial data publishing by role composition. Proc. VLDB Endow. 1(1), 845–856 (2008)

Detecting Inference Channels in Private Multimedia Data *via* Social Networks

Béchara Al Bouna and Richard Chbeir

LE2I Laboratory UMR-CNRS, University of Bourgogne
21078 Dijon Cedex France
{bechara.albouna,richard.chbeir}@u-bourgogne.fr

Abstract. Indirect access to protected information has been one of the key challenges facing the international community for the last decade. Providing techniques to control direct access to sensitive information remain insufficient against inference channels established when legitimate data reveal classified facts hidden from unauthorized users. Several techniques have been proposed in the literature to meet indirect access prevention. However, those addressing the inference problem when involving multimedia objects (images, audio, video, etc.) remain few and hold several drawbacks. In essence, the complex structure of multimedia objects makes the fact of detecting indirect access a difficult task. In this paper, we propose a novel approach to detect possible inference channels established between multimedia objects representing persons by combining social network information with unmasked content of multimedia objects. Here, we present the techniques used to map the content of social networks to the set of multimedia objects at hand. We also provide an `MiD` function able to determine whether an unmasked multimedia object combined with data from the social network infers a sensitive multimedia object.

Keywords: Inference Channels, Multimedia, Access Control.

1 Introduction

Providing appropriate techniques to protect sensitive information to be published and shared requires both 1) defining direct access and who has the right to perform a specified operation on confidential resources, and 2) preventing indirect access to information occurring when legitimate data reveal classified facts hidden from unauthorized users. On one hand, several access control models [9] [15] [18] have been proposed in the literature to meet direct access prevention requirements. Recently, with the increased use of multimedia objects (images, audio, video, etc.), the tradeoff between data availability and privacy has lead to the definition of adapted models [1] [4] [5] [6] [7] providing safe browsing and publishing of multimedia objects' contents. Particularly, masking out objects of interests representing persons is of great importance in several privacy scenarios (e.g. hiding the face of a popular person in a TV show). On the other hand, several studies [10] [12] [20] [27] [29] have focused on handling various forms of indirect access commonly known as the inference problem. They focus mainly on preventing inference channels in textual-based applications.

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 208–224, 2009.

However, none to our knowledge has explored the damage that might be caused by inference channels established in multimedia-based environments due to social networks or other common knowledge. In fact, social networks are becoming popular[1] and attracting lots of people and organizations who publish data, pictures, and share visions, ideas, hobbies, friendship, kinship, dislike, etc. Almost every user has an account on a social network with information containing pictures of him and a set of relations established with others (*friendOf, CollegeOf, inRelationshipWith*, etc.). In many situations, such information or common knowledge combined with unmasked content of multimedia objects make high the potential risk of uncovering the sensitive content of multimedia objects.

In this paper, we address privacy protection in multimedia objects representing persons by detecting inference channels thanks to knowledge gathered from social networks. Our study aims to detect whether a masked content of multimedia objects is endangered due to combining the social networks knowledge with unmasked (salient) objects. Here, we propose a two-phase approach to elaborate, on one hand, the social networks knowledge representation and multimedia objects mappings, and to provide, on the other hand, algorithms to detect possible inference channels established when protecting one or several sensitive multimedia objects. To the best of our knowledge, this is the first study to address multimedia-based inference problem using social networks.

The rest of this paper is organized as follows. In Section 2, we describe a motivation scenario to show the risks rose from social networks when protecting multimedia objects' content. In Section 3, we point out the set of techniques proposed in the literature to tackle inference channels. In Section 4, we present a set of definitions needed to fully understand our approach. In Section 5, we present our proposal holding a mapping module to map social network nodes and edges to multimedia objects. Finally, we conclude our paper and present some future directions.

2 Motivating Scenario

Let us consider a company holding a local image database accessible to all the staff members, visitors, trainees, clients, and collaborators. The images are categorized as follows:

- *Social dinner events*: containing all the photos of the staff members taken during social dinners with their husbands (or wives) and relatives.
- *Meetings*: containing all the images taken during meetings of staff members held in the research department.

To manage these images, a package containing a set of functions (distortion, face detection, movement detection, etc.) is provided with a search engine allowing to retrieve images using query-by-example techniques based-on low-level features (colors, texture, shapes, etc.), similar object (i.e. sample image), or meta-data (keywords). A simple publication policy is defined in the company to preserve privacy ethics when publishing the content of the database; it states that *all staff members who are part of the head office of the research department should not be appear in social events' photos*. In

[1] For example, the Facebook social network holds more than one hundred million users.

order to apply this policy, the webmaster, after identifying (automatically and/or manually) the related images, uses a blur filter function to hide the related multimedia objects content. Fig. 1 shows both a photo of Mr. Dupond, head of the research department, with his wife and colleagues, taken in one of the social dinners, and the same photo after applying the publication policy where his face was blurred. Mr. Dupond is also active on the web and has an account on a known social network where he posts and shares several information with his friends, family and wife. In this situation, one can see that hiding the face of Mr. Dupond won't be enough here. That is, people who have access to information at the social network and aware of the identity of his wife, might easily recognise him *via* the presence of his wife sited nearby in the image to be secured.

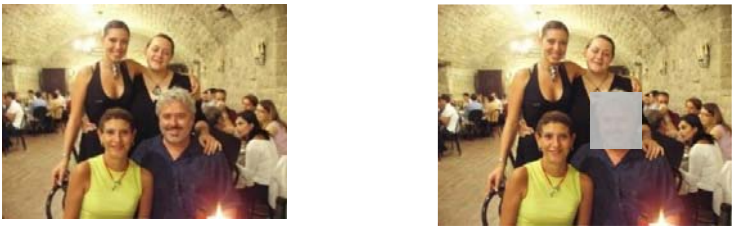


Fig. 1. Social dinner photo with the head of the research department, Mr. Dupond, before and after applying the publication policy

This type of inference problem, that we call *inference by domain knowledge*, involving multimedia objects remains critical. In essence, combining unmasked information with the knowledge of the application domain might reveal interesting information which puts privacy at risks. However, as mentioned before, none has considered its influence when protecting sensitive content of multimedia objects. Our work here is dedicated to suggest a challenging solution.

3 Related Work

In this section, we present an overview of the studies conducted in the literature to address inference detection and elimination techniques in three different areas: database, XML, and multimedia environments.

The inference problem in databases occurs when sensitive information can be disclosed from non sensitive data combined with either metadata/database constraints or external data related to the domain knowledge. Such an issue has been widely discussed in the database environment where users are able to establish inference channels based on the knowledge extracted from the domain. In [20], the authors use classical information theory to calculate the bandwidth of illegal information flow using an INFER function. An inference channel is established if there exists an item in the *sphere of influence* which is composed of entities, attributes, relationships and constraints with a classification higher than the classification of specified information. Research led by Hinke and Delugach in [12] led to the definition of Wizard [13], a

system that takes a database schema as input and tests if it is possible to establish inference channels according to a set of predefined semantic graphs related to the domain knowledge. In this approach, domain knowledge data are acquired in a micro-analysis to enrich database semantics. In [10], the authors present a Semantic Inference Model (SIM) based on the data contained in a given database, the schema of the database, and the semantic relations that might exist between the data. The authors use Bayesian Networks in order to calculate the possibility of inferring sensitive information.

Several studies have emerged to tackle indirect access caused by inference channels in XML environments. The work led by Yang and Li in [28] proposes an interesting approach in which it is possible to detect inference channels established when combining common knowledge with unclassified information along with others related to functional dependencies between different nodes in an XML document. Their approach is based on *conditions* $\rightarrow$ *facts* in order to represent XML constraints. The authors use an algorithm to construct an AND/OR graph which helps removing unnecessary links between unclassified information and the sensitive ones.

The described approaches are interesting and provide satisfactory results (depending on the application domain) when handling textual data. However, they cope badly with multimedia data. In fact, detecting inference channels in multimedia environments still complex for two main reasons: 1) the semantic gap between the low level features and the semantic meaning of a multimedia object, and 2) the complex structure of multimedia objects. Few studies have addressed so far the effect of inference when controlling multimedia objects. In [14], the authors define an interesting approach to replace salient objects of a video with virtual objects. Although, the approach looks efficient in preserving privacy and eliminating statistical inference, it puts however at risk data semantics where it becomes difficult to recognize some crucial content of the video.

4 Preliminaries and Definitions

In this section, we define the main concepts on which our approach relies. We first describe the basic concepts of multimedia objects, sensitive multimedia objects, multimedia relations, similarity functions and domain relations. After, we give the formal representation of social networks and show how it is possible to enrich social networks with new knowledge using explicit and user-defined rules.

Definition 1 – Multimedia Object (`MO`). Represents any type of multimedia data such as text, image, video, or a salient object describing an object of interest (e.g. face of a person.). It is formally represented in our approach as:

$$\texttt{MO}: \langle \texttt{id}, \texttt{A}, \texttt{O} \rangle$$

where:

- `id`: is the identifier of the multimedia object
- `A`: represents the set of textual attributes describing the multimedia object. It is formally defined as: $\langle a_1:val_1, \ldots, a_n:val_n \rangle$ where each a_i represents an element

in the of Dublin Core Metadata Element set[2] (source, description, date, contributor, format, etc.), MPEG-7 semantic set[3] (semantic place, concept, state, event, object, etc.), or any keywords

- O: contains the raw data, the link, or a representation that characterizes the multimedia object. It is formally defined as: $\langle o_1:val_1,\ldots,o_n:val_n\rangle$ where o_i can be a BFILE, an URL/URI, or an URL/URL augmented with a primitive to represent the object (e.g. Minimum Bounding Rectangle, Circle, etc.).

Fig. 2 shows an extract of the description of multimedia objects in Fig. 1 using our multimedia type representation. For the sake of simplicity, we represent in the following a multimedia object having an identifier i as mo_i.

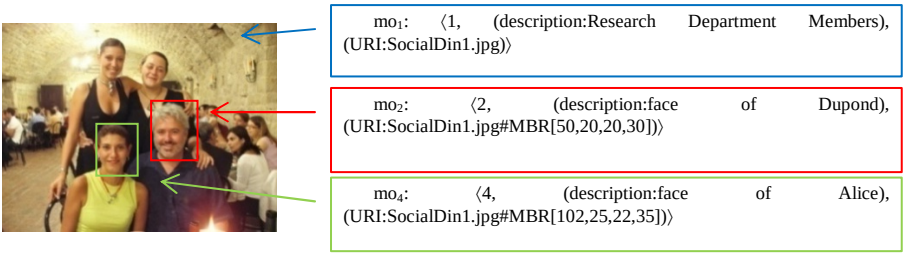


Fig. 2. Description of the Social Dinner photo with the head of the research department

Definition 2 – Sensitive Multimedia Object (SMO). Is a multimedia object to be protected from unauthorized users. It is formally described as:

$$SMo: \langle Mo, C_f\rangle$$

where:

- Mo is (the identifier of) the multimedia object to be protected
- C_f refers to one or several multimedia protection function(s) (blur filter function, mosaic filter function, spiral filter function, substitution function, etc.). Each function can have its input parameters (e.g. the blur filter has a mosaic filter level varying from 1 to 10 (as defined in [8]). Details about C_f are omitted here due to the lack of space. For instance, hiding the identity of Mr. Dupond in Fig. 1 can be described as smo_2: $\langle mo_2, Blur(7)\rangle$

Definition 3 – Multimedia Relation (MR). Represents a predefined multimedia relation that can link a set of multimedia objects and can be generated (automatically) using low-level features (shape, location, etc.). Each MR can be formally defined as:

$$MR: \langle name,\ type,\ P\rangle$$

[2] http://dublincore.org/documents/dcmi-terms/
[3] http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

where:

- `name` is the name used to identify the relation
- `type` ∈ {co-occurrence, topologic[4], directional[5], temporal[6], metric[7], semantic[8]}
- `P` ⊆ {reflexive, symmetric, transitive, associative**}** is a set of properties that characterize the multimedia relation

A `MR` is instantiated in our approach as a statement of the following form:

$$\texttt{MR.name}(mo_1, \ldots, mo_n)$$

For instance, the face of `Mr. Dupond` located to the left of `Alice`'s in SocialDin1.jpg can be represented with $\texttt{Left}(mo_2, mo_4)$.

Definition 4 – Similarity (S). Is used to compare and measure the similarity between either textual descriptions or multimedia features. It is defined as:

$$S(X, Y) = \langle f_1(\langle x_1, \ldots, x_n\rangle, \langle y_1, \ldots, y_m\rangle), \ldots, f_k(\langle x_1, \ldots, x_n\rangle, \langle y_1, \ldots, y_m\rangle)\rangle = \langle \delta_1, \ldots, \delta_k\rangle \;/\; n, m, k \in \mathbb{N}$$

where:

- $x_i \subseteq X$ and $y_j \subseteq Y$ represent the set of terms/expressions/features or multimedia objects to be compared depending on the similarity functions used
- f_i is either a set of textual similarity functions (edit distance, n-grams, etc.) or multimedia similarity functions[9]
- $\langle \delta_1, \ldots, \delta_n\rangle$ is the vector of scores returned by the similarity functions $\langle f_1, \ldots, f_n\rangle$ where $\delta_i \in [0, 1]$.

In the following, S_T and S_M will be used to designate Textual Similarity and Multimedia Similarity respectively.

Let us illustrate this, for instance, by computing the multimedia similarity between the picture of `Mr. Dupond` in Fig. 3 (represented as mo_{20}), and the photo of social event in Fig. 1 using the following multimedia functions f_1 and f_2:

- f_1 is related to the InterMedia Oracle module [21] based on color segments to calculate image similarity

[4] Such as Disjoint, Touch, Overlap, Equal, Contain, Inside, Cover, and isCoveredBy.

[5] Such as North, South, East, West, North-south, Northwest, Southeast, Southwest, Left, Right, High, Below, In front of, and Behind

[6] Such as Before, After, Touches, Overlaps, BeginsWith, EndsWith, Contains, During, and Equal

[7] Such as Far, Close, etc.

[8] It can represent what it is fixed by the nature (e.g. InLoveWith) or describe a social relationship (e.g. isMarriedTo), etc.

[9] Several multimedia functions are provided in the literature. For instance, some DBMSs such as Oracle and DB2 provide SQL-operators [16] while others are accessible via API functions [17] and web services [3]. Details on such functions and their applications are out of the scope of this paper.

- f_2 is based on color object recognition and SVM classifiers. It computes decisions based on a set of classes representing the trained images (See [26] for more details)

The obtained multimedia similarity has the following scoring set:

$$S_M(mo_{20}, mo_1): \langle f_1(mo_{20}, mo_1), f_2(mo_{20}, mo_1)\rangle = \langle 0.6, 0.8\rangle$$

mo_{20}: ⟨20, (description:profile picture of Dupond, owner:Dupond), (URI:Pic1.jpg)⟩

Fig. 3. Profile Picture of Mr. Dupond

Similarly, to compute textual similarity, two different functions f_3 and f_4 are used where:

- f_3 is a string similarity function based on the Levenshtein edit distance [19]
- f_4 is based on the number of different trigrams existing in the input text [19].

The obtained textual similarity has the following scoring set:

$$S_T(\text{“face of Dupond”}, \text{“Dupond”}): \langle f_3(\text{“face of Dupond”}, \text{“Dupond”}), f_4(\text{“face of Dupond”}, \text{“Dupond”})\rangle = \langle 0.11, 0.2\rangle^{10}$$

Definition 5 – Aggregation Function (μ). Aggregates a set of values (returned by a similarity *S*) in order to select or compute one value to be considered. As several similarity functions can be used to compute the similarity between either textual-based or multimedia-based features, it is important to retrieve the most appropriate result for a given situation so to facilitate decision-making. An aggregation function can be defined by classical aggregation function (average, minimum, maximum, etc.) or any probabilistic function (the combination rule of Dempster and Shafer theory of evidence (DS) [22] [25], Bayesian Decision theory [23], Decision Trees [24], etc.). More details on aggregation functions can be found in [2]. It is formally written in our approach as:

$$\mu(S, \varepsilon) = \beta \in [0,1]$$

where:

- **S** is a textual or multimedia similarity (as defined in Def. 4)
- **ε** is an uncertainty threshold belonging to the interval [0,1]. It represents the percentage of uncertainty related to the combination of similarity functions used in

[10] Here we compare both sentences; however other techniques could be more precise to compute string similarity such as finding whether a sentence is contained in another, or compare ng individual words, etc.

the related similarity. In fact, ε can affect the overall scores returned by individual S_T or S_M. If omitted, ε = 0

- β is the (normalized) aggregated score.

For instance, by applying the average aggregation function on the result set obtained by f_1 and f_2 when comparing the picture of `Mr. Dupond` in Fig. 3 and the photo of social event in Fig. 1, we obtain the aggregated score of `β = (0.6 + 0.8)/(2+0.1) = 0.67` (after having assigned 0.1 to ε related to S_M).

Definition 6 – Domain Entity (DE). Represents a user, group, project, or organization in a social network. A **DE** can be formally described as:

$$DE: \langle id, name, CRED, MO \rangle$$

where:

- `id` is a unique identifier
- `name` is the name describing the domain entity (e.g. `Dupond`)
- `CRED` is a set of credentials which characterize **DE** in the social network**.** It is formally defined as*:* $\langle cred_1:val_1, \ldots, cred_n:val_n \rangle$ where each $\mathbf{cred_i}$ can represent common element in FOAF[11] (name, phone, email, etc.), SIOC[12] (about, resource, etc.), or other social network descriptors. For instance, it could be `location:France, University:Dijon,` etc.
- `MO` represents a set of multimedia objects describing the **DE**.

For instance, the **DE** describing the profile of the user `Dupond` can be described as:

$$de_1: \langle 1, Dupond, \langle location:France \rangle, \langle mo_{20} \rangle \rangle.$$

In the following and for the sake of simplicity, a **DE** having *name* and an identifier *i* will be referenced as $name_i$.

Definition 7 – Domain Relation (DR). Represents an application domain-related and/or semantic relation. A **DR** can be formally defined as:

$$DR: \langle name, type, P, Exp \rangle$$

where:

- `name` is the name used to identify the relation
- `type` ∈ {ontologic[13], semantic}
- `P` ⊆ {reflexive, symmetric, transitive, associative} is a set of properties that characterize the relation

[11] `http://www.w3.org/2001/sw/Europe/events/foaf-galway/`
[12] `http://www.w3.org/2008/09/msnws/papers/sioc.html`
[13] `isA, instanceOf, etc.`

- Exp is a Boolean expression used to represent (when possible) the designated relation throughout a set of **MR.** For instance, IsSittingNear.Exp = Left ∨ Right ∨ Above ∨ Below.

5 Proposal

The problem of determining the amount of information leakage in a set of unmasked multimedia objects MMDB is mainly related to both the representation of the application domain knowledge at hand, and the semantic gap existing between low-level features and the meaning of multimedia objects. In essence, in order to protect sensitive multimedia objects SMO contained in MMDB, one should be able to identify possible correspondences between SMO and some unmasked multimedia objects existing in MMDB through a common knowledge (to be extracted here from social networks).

To address these issues, we provide here an approach composed of two main levels holding, on one hand, information gathered from social networks and, on the other hand, the set of multimedia objects in MMDB. It includes:

1. A rich and flexible representation of social networks formally described as a Domain Knowledge (D_K) able to consider common features and multimedia descriptions and standards.
2. A framework dedicated to detect multimedia-based inference channels bearing three main modules:
   a. A *Mapping Module* (MpP): allowing to map the social networks content to the set of multimedia objects MMDB
   b. An *Inference Detection Module* (IDM): allowing to detect inference channels and to determine the amount of information leakage related to SMO
   c. An *Inference Elimination Module* (IEM): able to filter out all the inference channels detected.

In the following, we will detail our proposal components and discuss the process of detecting inference channels. The Inference Elimination Module will be detailed in another dedicated study.

5.1 Domain Knowledge (D_K)

In our approach, a Domain Knowledge (D_K) is used to organize the nodes and their relationships in social networks at hand into a semantic graph. It is formally defined as:

$$D_K: \langle N, E, W, \nu \rangle$$

where:

- N is the set of nodes representing users, groups, projects, and organizations in a social network. Each node $n \in DE$
- E is a set of edges interconnecting nodes of the social networks. An edge $e_i \in DR$. In the following, $e_i(n_1, n_2)$ and $e_i.name(n_1, n_2)$ are used interchangeably
- W is a set of values belonging to the interval [0,1]

- ν is a function assigning to each edge $e_i \in E$ a weight w_i, $\nu: E \rightarrow W$ so to reflect the importance of a corresponding relation on the social network.

In Fig. 4, we provide a graphical representation of an extract of the social network of Mr. Dupond in our running example. We do not detail here the process of transforming a social network into our representation D_K as it is straightforward and application-based.

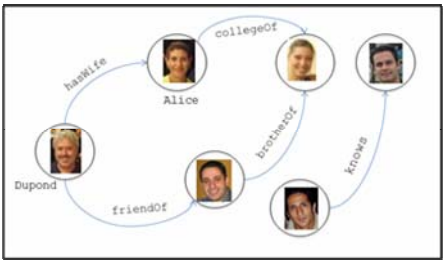


Fig. 4. Graphical representation of an extract of the social network of Mr. Dupond

In order to enrich the Domain Knowledge (D_K) with inferred semantics, we use a set of rules (Rg) representing derivation axioms. Each rule is defined as follows:

$$Rg: \text{antecedent} \rightarrow \text{consequent}$$

where:

- antecedent is the body of the rule. It is formed by a set of conjunct atoms of DR relations between variable nodes written as antecedent= $dr_1(a, b)\ op\ \ldots op\ dr_n(x, y)$ where op $\in \{\wedge, \vee, \neg,$ etc.$\}$ and a, b, x, y represent variables or instances of D_K
- consequent is a DR relation representing the head of the rule.

For instance, Rg_1: FriendOf(n_1,n_2) $\wedge$ MarriedTo(n_2,n_3) $\rightarrow$ Knows (n_1,n_3) states that if a node $n_1 \in N$ is a friend with a node $n_2 \in N$, then the former must know the spouse of the later.

5.2 Inference Framework

In this section we present our inference framework formed by a mapping module and an inference detection module.

5.2.1 Mapping Module (MpP)

In order to identify the correspondence between D_K content and multimedia objects in MMDB, two different but related mappings are used:

- *Node Mapping* (M_N): capable of identifying the correspondence between nodes of D_K and multimedia objects in MMDB
- *Edge Mapping* (M_E): represents the process of checking whether the edge is valid at the MMDB level according to the related DR.Exp defined.

5.2.1.1 Node Mapping (M_N). In our approach, mapping nodes in D_K to MMDB considers the multi-criteria aspect of multimedia objects and descriptors by matching related low-level and textual features. We formally describe the node mapping M_N as:

$$M_N(mo, n) = \mu(S(X, Y), \varepsilon) \rightarrow \alpha$$

where:

- $mo \in MMDB$ is a multimedia object to be mapped
- $n \in N$ is a node of D_K
- S is the similarity between X and Y where $X \subseteq MO$ and $Y \subseteq N$
- μ represents an aggregation function[14] with its related uncertainty threshold ε used to aggregate the set of returned scores by S. The aggregated score returned by μ is compared to α in order to raise or not a mapping between the mo and n.
- $\alpha \in [0,1]$ is a the returned result of the node mapping process

For instance, in order to automatically compute the following mappings[15]:

$M_N(smo_2, Dupond_1) = Max(Avg(S_T(smo_2.A, Dupond_1.n\_name), 0.1), DS(S_M(smo_2.O, Dupond1.mo_{20}.O), 0.1)) = Max(Avg(\langle 0.2, 0.111\rangle, 0.1), DS(\langle 0.8, 0.6\rangle, 0.1)) = Max(0.14, 0.78) = 0.78$

$M_N(mo_4, Alice_2) = Max(Avg(S_T(mo_4.A, Alice_2.n\_name), 0.1), DS(S_M(mo_4.O, Alice_2.mo_{40}.O), 0.1)) = Max(Avg(\langle 0.2, 0.111\rangle, 0.1), DS(\langle 0.8, 0.7\rangle, 0.1)) = Max(0.14, 0.84) = 0.84$

The algorithm of node mapping (M_N) to MMDB is given below:

Algorithm 1. [**MO2N_Mapping**]	**Line**
Input: MMDB, D_N, DR /* MMDB *is* the set of multimedia objects, D_K is the domain knowledge DR is a specified set of domain relations */	1
Output: Map_Score(N, MMDB) // a matrix with score related to nodes mapped to MMDB	
Begin	
For each mo_i **in** MMDB **Do**	4
$n_0 = D_K(0)$ // represent a chosen node from the D_K	
Map_Score ← DFS(n_0, mo_i, *score*, *DR*)	
End For	
Return Map_Score	
End	9

Algorithm 1 establishes a mapping between a set of multimedia objects MMDB and their nodes N interconnected using the set of edges DR of the domain knowledge. And

[14] Different aggregation functions can be assigned according to the similarity functions used. That is, we could assign an aggregation function to compute values returned by textual similarity functions and Bayesian networks to compute values returned by multimedia similarity functions.

[15] Details about computation are omitted here.

so, a matrix holding mapping scores[16] of multimedia objects and nodes is retrieved. Algorithm 2 is used to search the D_K graph using the Depth First or DFS algorithm [11] to retrieve a vector of scores related to the mapping of a multimedia object mo_i to the set of nodes of D_K using node mapping M_N.

Algorithm 2. [DFS]	**Line**
Input:n, mo, score, DR /*n is the node to map, mo is a multimedia object ∈ MMDB, score represents the vector to hold the mapping scores between n and mo */	1
Output: score(n, mo)	
Begin	
score ← M_N(n, mo)	4
For each n_i **such that** (n,n_i) **is an edge** e_i **in DR Do**	
IF n_i **was not visited yet THEN**	
Dfs(n_i, mo, score, DR)	
End For	
Return score	
End	10

5.2.1.2 Edge Mapping (M_E). Mapping edges refers to the process of finding whether any edge e_i defined at D_K level has a valid description at the MMDB level. Also, it is used to validate the expression defined for each of the DRs at D_K level. We formally define the edge mapping M_E as follows:

$$M_E(e_i, mo_n, mo_m) = g(e_i.Exp, mo_n, mo_m)$$

where:

- e_i is an edge interconnecting two different nodes in D_K
- mo_n and mo_m are two different multimedia objects in MMDB
- *g* is a Boolean function able to evaluate the expression e_i.Exp (i.e., the Boolean expression defined in e_i) with respect to (w.r.t.) mo_n and mo_m

In other words, an edge e_i, representing an DR in D_K, is mapped to MMDB if ∃ mo_n and mo_m that validate the set of multimedia relations MR contained in the Boolean expression Exp of e_i. For instance, the hasWife holding a Boolean expression hasWife.Exp = Left ∨ Right is mapped to MMDB if Left(mo_i, mo_j) ∨ Right(mo_i, mo_j) are valid for the existing multimedia objects mo_i and mo_j ∈ MMDB. In the following, we describe our inference detection module defined to determine possible inference channels.

5.2.2 Inference Detection Module (IDM)

To detect inference channels, we define a *Multimedia based inference detection function* called MiD to detect the possible risk of inferring a sensitive multimedia object smo_m from a given multimedia object mo_n according to their corresponding mapped

[16] Mapping nodes to their corresponding multimedia objects should be performed in the preprocessing phase due to the heavy computation time needed for processing semantic similarity between multimedia objects.

nodes n_1 and n_2 at D_K level. Our MiD can detect if the multimedia object mo_n infers the sensitive multimedia object smo_m w.r.t. the nodes n_1 and n_2 as:

$$MiD(mo_n \rightarrow smo_m)_{(n1,n2)} = \mu(\langle M_N(mo_n, n_1),\ M_N(smo_m, n_2)\rangle, \varepsilon) \times \psi((n_1, n_2))_{(mo_n, smo_m)} > \gamma$$

where:

- $mo_n, smo_m \in MMDB$, $n_1, n_2 \in N$, and $e_l \in E$ linking n_1 to n_2
- $M_N(mo_n, n_1)$ and $M_N(smo_m, n_2)$ represent the scores related to the mapping between n_1 and n_2 and their corresponding multimedia objects mo_n and sensitive multimedia objects smo_m respectively.
- $\psi((n_1,n_2))_{(mo_n,smo_m)}$ is a function that returns a value representing the maximum computed weight of the set of DR between n_1 and n_2, i.e. $\psi((n_1,n_2))_{(mo_n,smo_m)} = Max\ (\cup^{n}_{l=1} M_E(e_l, mo_n, smo_m) \times e_l.w)$. Max could be replaced by any other aggregation function (see Def. 5) w.r.t. the domain of application. l represents the number of relations existing between the nodes n_1 and n_2. These set of relations are either directly related or inferred (w.r.t. both the set of properties P, such as symmetric, transitivity, predefined for each relation), and the predefined explicit rules Rg used to enrich the D_K. We consider that, an edge e_l between two nodes n_1 and n_2 could provide potential inference at the MMDB level independently from the mapping direction and its symmetric property. That is, if both mo_n and smo_m are mapped to n_1 and n_2 respectively, an edge e_l between n_1 and n_2, could be considered as possible threat whether it is defined as $e_l(n_1, n_2)$ or $e_l(n_2, n_1)$ unless the edge mapping M_E with mo_n and smo_m has computed a false value. For example, the relation $hasWife(Dupond_1, Alice_2)$ between the nodes $Dupond_1$ and $Alice_2$ provides potential inference knowing that, on one hand, mo_4 (the multimedia object representing Alice) is mapped to the node $Alice_2$ and the smo_2 (the multimedia object representing Mr. Dupond) is mapped to the instance $Dupond_1$, and, on the other hand, the relation mapping $M_E(mo_4, smo_2)_{hasWife}$ is valid.
- γ represents a predefined threshold varying between [0, 1] on which MiD is based to determine whether the multimedia object mo_n infers the sensitive multimedia object smo_m

An smo_m is considered *safe* if its corresponding mapped nodes at D_K level have no upward and downward edges that could be discovered at the MMDB level leading consequently to its identification. Formally:

$\forall\ mo_n, smo_m \in MMDB, n_1, n_2 \in N$, and $e \in E$, smo_m is safe $\Rightarrow \nexists\ M_E(mo_j, mo_n, e) = 1$, $M_N(mo_j, n_1)$, $M_N(smo_m, n_2)$ and $MiD(mo_n \rightarrow smo_m)_{(n_1, n_2)} > \gamma$

which means that an inference channel could be established in a multimedia environment between $mo_n, smo_m \in MMDB$ if their mapped nodes n_1 and n_2 respectively,

are related at the social network level and the social network dependent relation e_k between n_1 and n_2 is mapped to the set MMDB.

In order to illustrate the use of MiD function, we will refer to our motivating scenario. To hide the face of Mr. Dupond (described using smo_2), one should check to see if the DR hasWife between $Dupond_1$ and $Alice_2$ could lead to the identification of Mr. Dupond. Both nodes $Dupond_1$ and $Alice_2$ are mapped to the MMDB and there exists an edge mapping that could return true for the mapped multimedia objects where the edge e_k is defined as $\langle hasWife, 0.7\rangle$. $w=0.7$ represents a weight reflecting the relevance of the DR at D_K level. If we wish to protect the multimedia object smo_2 representing the face of Mr. Dupond, let us determine now the possible threat due to the multimedia object mo_4. In this case, the MiD function is defined as follows:

$$MiD(mo_4 \rightarrow smo_2)_{(Alice_2,\ Dupond_1)}$$
$$= Avg\ (\langle M_N(mo_4, Alice_2), M_N(smo_2, Dupond_1)\rangle, 0.0) \times \psi(Alice_2, Dupond_1)_{(mo_4, smo_2)} > 0.5$$

The mappings are as follows:

$M_N(smo_2,\ Dupond_1) = 0.78$ and $M_N(mo_4,\ Alice_2) = 0.84$.
$\psi(Alice_2, Dupond_1)_{(mo_4, smo_2)} = M_E(mo_4,\ smo_2)_{hasWife} \times 0.7$.

Both multimedia objects mo_4 and smo_2 are mapped to nodes in D_K. Furthermore, the nodes $Dupond_1$ and $Alice_2$ are related with the edge e_k representing the hasWife DR. As mo_4 is located to the left of smo_2 in the same image SocialDin1.jpg $\in$ MMDB, the edge mapping of hasWife defined as $M_E(mo_4,\ smo_2)_{hasWife}$ is satisfied. Finally, $\psi(Alice_2, Dupond_1)_{(mo_4, smo_2)} = 0.7$ as there are no other edges between nodes $Dupond_1$ and $Alice_2$ in this example. Thus, the final result computed by MiD is: $MiD(mo_4 \rightarrow smo_2)_{(Alice2,\ Dupond1)} = Avg(\langle 0.78,\ 0.84\rangle,\ 0.0) \times 0.7 = 0.567$. This means that the multimedia object mo_4 infers the sensitive multimedia object smo_2 as the result returned by the MiD is greater than the predefined threshold 0.5. Algorithm 3 is used to highlight threatening multimedia objects that might lead to the identification of a sensitive multimedia object smo.

The Inference Detection algorithm works as follows. First, we retrieve the set of nodes mapped to the sensitive multimedia object smo from the matrix Map_Score(N, MMDB) computed previously. For each node n_i within the retrieved set, we get its adjacent nodes according to the specified edges E at D_K level. We consider two different types of related nodes: *directly* related nodes (i.e. $hasWife(Dupond_1,\ Alice_2)$), and *inferred* nodes using either D_K rules or implicit relation-based rules (based on their properties i.e. isRelatedTo(a,c), isRelatedTo(c,b) $\Rightarrow$ isRelatedTo(a,b) when isRelatedTo is transitive). For each node in the set of adjacent nodes related to node n_i, we retrieve the corresponding multimedia objects using the retrieveMO function, according to the set of mappings already computed. We determine consequently whether a multimedia object mo_k related to smo is a possible threat which is determined using the predefined MiD function. That is, a multimedia object is considered threatening if the value returned by the MiD function is greater than the input value γ. The final computed result represents a set of Threatening Multimedia Objects TMO.

Algorithm 3 [**Inference_Detection]** Line

Input: smo, γ, Map_Score(N, MMDB), E /*smo represents the sensitive mo, γ is the inference threshold, Map_Score is the mapping matrix between nodes and MMDB, E is the predefined set of edge to consider while detecting relations between nodes */ 1

Output: TMO // a set of threatening multimedia objects

Begin

N = retrieveNodes (smo, Map_Score (N, MMDB)) 4
// retrieve the nodes mapped to smo from the Map_Score matrix

For each n_i **In** N

$AdjN_D$ = getDirectlyRelatedNodes (n_i, E)
//retrieve all nodes directly related to the n_i according to the edges in E

$AdjN_I$ = getInferredNodes(n_j) //retrieve all nodes inferred from either an explicit rules
// (user defined) or implicit rules (according to predefined relation properties)

$AdjN = AdjN_D \cup AdjN_I$

For each n_j **In** AdjN

MO = retrieveMO (n_j, Map_Score (N, MO)) // retrieve the multimedia objects
// mapped to n_j from the Map_Score matrix

For each mo_k **In** MO

$t = MiD(mo_k \rightarrow smo)_{(ni, nj)}$

If ($t > \gamma$)

$TMO \leftarrow mo_k$

End If

End For

End For

End For

Return TMO

End 20

6 Conclusion and Future Work

In this paper, we proposed a technique to protect privacy from inference channels established in a multimedia environment by combining social networks information with unmasked multimedia objects content. Our approach is based on a generic domain knowledge in which we describe nodes and edges representing the social network data. We also proposed techniques to map these data to the set of multimedia objects to be protected. A MiD function is used to detect whether a multimedia object mo_i infers a multimedia object mo_j according to the mapped nodes and relations.

In the future work, we intent to test the efficiency of our MiD function w.r.t. different multimedia and textual mapping techniques. We further wish to tackle inference related to the returned result from multiple queries which could lead to uncovering sensitive multimedia objects.

References

1. Adam, N.R., Atluri, V., Bertino, E., Ferrari, E.: A Content Based Authorization Model for Digital Libraries. IEEE Transaction on Knowledge And Data Engineering, 296–315 (2002)
2. AL Bouna, B., Chbeir, R., Miteran, J.: MCA2CM: Multimedia Context-Aware Access Control Model. In: Intelligence and Security Informatics, pp. 115–123. IEEE, New Brunswick (2007)

3. Alliance, ASP. (s.d.), http://aspalliance.com/404_Image_Web_Service (visited: Feburary 16, 2008)
4. Atluri, V., Chun, S.A.: An Authorization Model for Geospatial Data. IEEE Tansaction on Dependable And Secure Computing 1(4), 238–254 (2004)
5. Bertino, E., Fan, J., Ferrari, E., Hacid, M.-S., Elmagarmid, K.A., Zhu, X.: A Hierarchical Access Control Model for Video Database Systems. ACM Trans. Inf. Syst., 155–191 (2003)
6. Bertino, E., Ferrari, E., Perego, A.: Max: An Access Control System for Digital Libraries and the Web. In: COMPSAC, pp. 945–950 (2002)
7. Bertino, E., Hammad, M.A., Aref, W.G., Elmagarmid, A.K.: Access Control Model for Video Databases. In: 9th International Conference on Information Knowledge Management, CIKM, pp. 336–343 (2000)
8. Boyle, M., Edwards, C., Greenberg, S.: The effects of filtered video on awareness and privacy. In: CSCW, pp. 1–10. ACM, Philadelphia (2000)
9. Chamberlin, D.D., Gray, J., Irving, L.T.: Views, Authorization, and Locking in a Relational Database System. In: ACM National Computer Conference, pp. 425–430 (1975)
10. Chen, Y., Chu, W.W.: Protection of Database Security via Collaborative Inference Detection. IEEE Transactions on Knowledge and Data Engineering (TKDE), Special Issue on Knowledge and Data Management and Engineering in Intelligence and Security Informatics (2007)
11. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Depth-first search. In: Dans Introduction to Algorithms, pp. 540–549. MIT Press and McGraw-Hill (2001)
12. Delugach, H.S., Hinke, T.H.: Using Conceptual Graphs To Represent Database Inference Security Analysis. Jour. Computing and Info. Tech. 2(4), 291–307 (1994)
13. Delugach, H.S., Hinke, T.H.: Wizard: A Database Inference Analysis and Detection System. IEEE Transactions on Knowledge and Data Engineering 8, 56–66 (1996)
14. Fan, J., Luo, H., Hacid, M.-S., Bertino, E.: A novel approach for privacy-preserving video sharing. In: CIKM, pp. 609–616. ACM, Bremen (2005)
15. Ferraiolo, D.F., Barkley, J.F., Khun, D.R.: A Role-Based Access Control Model and Reference Implementation within a Corporate Intranet. ACM Transactions on Information and System Security (TISSEC) (2), 34–64 (1999)
16. IBM. (s.d.). QBIC - DB2 Image Extenders, http://wwwqbic.almaden.ibm.com (visited : February 16, 2008)
17. Lab, e. C. (s.d.). Image Processing, http://www.efg2.com/Lab/Library/ImageProcessing/SoftwarePackages.htm (visited : January 2, 2008)
18. Landwehr, C.: Formal Models of Computer Security. ACM Computer Survey 13, 247–278 (1981)
19. Lin, D.: An Information-Theoretic Definition of Similarity. International Machine Learning Society, Madison (1998)
20. Morgenstern, M.: Controlling logical inference in multilevel database systems. In: IEEE Symp. on Security and Privacy, pp. 245–256. IEEE, Oakland (1988)
21. Network, O. T. (s.d.). Oracle Multimedia, http://www.oracle.com/technology/products/intermedia/index.html (visited : November 9, 2007)
22. Dempster, A.P.: A Generalization of the Bayesian Inference. Journal of Royal Statistical, 205–447 (1968)

23. Poole, D.: Logic, Knowledge Representation, and Bayesian Decision Theory. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS, vol. 1861, pp. 70–86. Springer, Heidelberg (2000)
24. Quinlan, J.R.: Induction of Decision Trees. Machine Learning 1(1) (1986)
25. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)
26. Smach, F., Lemaitre, C., Miteran, J., Gauthier, J.P., Abid, M.: Colour Object recognition combining Motion Descriptors, Zernike Moments and Support Vector Machine. In: IEEE Industrial Electronics, IECON, pp. 3238–3242. IEEE, Paris (2006)
27. Staddon, J.: Dynamic Inference Control. In: Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD), San Diego, California, USA, pp. 94–100 (2003)
28. Yang, X., Li, C.: Secure XML Publishing without Information Leakage in the Presence of Data Inference. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB), pp. 96–107. Morgan Kaufmann, Torronto (2004)
29. Yip, R.W., Levitt, K.N.: Data Level Inference Detection in Database Systems. In: IEEE Computer Security Foundations Workshop, pp. 179–189. IEEE, Rockport (1998)

Enforcing Confidentiality Constraints on Sensitive Databases with Lightweight Trusted Clients

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

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

Abstract. Existing approaches for protecting sensitive information stored (outsourced) at external "honest-but-curious" servers are typically based on an overlying layer of encryption that is applied on the whole information, or use a combination of fragmentation and encryption. The computational load imposed by encryption makes such approaches not suitable for scenarios with lightweight clients.

In this paper, we address this issue and propose a novel model for enforcing privacy requirements on the outsourced information which departs from encryption. The basic idea of our approach is to store a small portion of the data (just enough to break sensitive associations) on the client, which is trusted being under the data owner control, while storing the remaining information in clear form at the external (honest-but-curious) server. We model the problem and provide a solution for it aiming at minimizing the data stored at the client. We also illustrate the execution of queries on the fragmented information.

1 Introduction

The design of distributed databases and associated techniques have been a topic of interest in the 1970s, at the birth of relational technology. The scenario that was considered in those years was significantly different from the current scenario: the emphasis was on the implementation of systems owned by a large organization managing information systems in several centers that offered the opportunity for the processing of distributed queries. The current ICT scenario is instead characterized by many important opportunities for the use of distributed databases where previous assumptions do not hold. Two important differences compared to traditional approaches are: *1)* the need to integrate the services of database providers that do not belong to the same organization; *2)* the presence of a variety of platforms, with an increase in the number and availability of devices that have access to a network connection, together with the presence

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 225–239, 2009.

of powerful servers offering significant computational and storage resources. The first aspect forces the requirement to specify security functions limiting access to the information stored in the databases. The second aspect instead forces an environment where the data and computational tasks are carefully balanced between the lightweight device and a powerful remote server. The two aspects are strictly related, since the servers are typically owned by service providers offering levels of cost, availability, reliability and flexibility difficult to obtain from in-house operations. Note that we classify as "lightweight" any device that exhibits performance or storage features that are significantly worse than what can be offered for the specific application by a remote service provider. Mobile devices certainly fit this description, but the scenario can be extended to generic computational platforms.

The motivation of this work lies in the desire to define novel solutions for the processing of large data collections, which we assume managed by traditional database technology, in a scenario where we are interested in using the services of a honest-but-curious powerful server, with a robust guarantee that confidentiality of information is protected.

In the literature, this problem has been addressed by combining fragmentation and encryption, thus splitting sensitive information among two or more servers and encrypting information whenever necessary [1,8,9]. In [1], the sensitive relation is split into *two fragments* stored at *two non communicating servers*, which must not know the identity of each other. This limitation on the number of fragments produced implies that it is not always possible to protect the confidentiality of information by simply splitting the involved attributes into two fragments and therefore can be encrypted. The main limitations of this solution are that: *1)* the absence of communication between the two servers is clearly a strong assumption and difficult to enforce in real environments; *2)* the query evaluation process requires the data owner to interact with both servers for joining (if needed) the two fragments and to decrypt the attributes possibly encrypted that appear in the query. In [8,9] these limitations are removed since a relation R is split into *two or more fragments possibly stored on one server* and limits the use of encryption for protecting the single sensitive attributes. Furthermore, for query execution efficiency, attributes that are not represented in the clear within a fragment are represented in encrypted form, providing the nice property that each fragment completely represents the original relation. The consequence of this design choice is that to evaluate a query, it is sufficient to access a single fragment, thus avoiding join operations (needed with the previous paradigm), which are quite expensive. This solution however still requires the client to possibly decrypt the attributes appearing in encrypted form in the fragment for evaluating a condition on them or for returning them to the user.

A common assumption of these solutions is that encryption is an unavoidable price to be paid to protect the information. There are, however, situations where encryption may not be applicable for protecting information. One issue can be the computational load imposed by encryption. For instance, in systems more constrained in battery power rather than memory capacity, it is beneficial to

spend some memory space to save on power consumption. Also, in real systems keys can become compromized, and keys can become lost, making the protection of the system dependent on the quality of key management services, rather than on the quality and strength of the cryptographic functions. Since key management is known to be a difficult task, we expect that an encryption-less solution can be of interest for many important applications.

To address these situations, we propose a paradigm shift where information is protected without encryption. The basic idea is that a small portion of the sensitive information can be stored on the client, trusted for both managing the information and for releasing such a sensitive information only to the authorized users, while the remaining information can be stored on an external server. Obviously, from the information stored on the external server it should not be possible to reconstruct a sensitive association (confidentiality constraint) since otherwise a privacy violation occurs. Since we do not want to remove the assumption that the external servers are all honest-but-curious, the client is the only entity in the system that can manage a portion of sensitive data. Sensitive data and associations can then be protected by splitting the original relation R into two fragments, denoted F_o and F_s, stored at the client and at a honest-but-curious storage server, respectively.

In the following sections, we describe how to correctly apply the above-mentioned approach. The remainder of the paper is organized as follows. Section 2 presents the basic concepts of the model. Section 3 introduces the idea of a correct and minimal fragmentation. Section 4 analyzes the minimization problem. Section 5 discusses the execution of queries in this architecture. Section 6 presents related work. Finally, Sect. 7 draws some conclusions.

2 Basic Concepts

We consider a scenario where, consistently with other proposals (e.g., [1,8,10]), the data to be protected are represented with a single relation r over a relation schema $R(a_1,\ldots,a_n)$. We use the standard notations of the relational database model. Also, when clear from the context, we will use R to denote either the relation schema R or the set of attributes in R.

Privacy requirements are represented by *confidentiality constraints*, which express restrictions on the single or joint visibility (association) of attributes in R. Confidentiality constraints are formally defined as follows [1,8].

Definition 1 (Confidentiality constraint). *Let $R(a_1,\ldots,a_n)$ be a relation schema, a* confidentiality constraint *c over R is a subset of the attributes in R.*

While simple in its definition, the confidentiality constraint construct supports the definition of different privacy requirements that may need to be expressed. A singleton constraint states that the *values* assumed by an attribute are considered sensitive and therefore cannot be accessed by an external party. A non-singleton constraint states that the *association* among values of given attributes is sensitive and therefore should not be outsourced to an external party.

PATIENT

SSN	Name	DoB	ZIP	Job	Illness	Cause
123-45-6789	Alice	80/02/11	20051	Secretary	asthma	Dust allergy
987-65-4321	Bob	85/05/22	22034	Student	fracture	Car accident
147-85-2369	Carol	73/07/30	22039	Secretary	carpal tunnel	Secretary
963-85-2741	David	75/11/26	20051	Lawyer	hypertension	Stress
789-65-4123	Emma	90/03/15	22035	Student	asthma	Student
123-65-4789	Fred	68/08/07	22034	Accountant	hypertension	Wrong diet

(a)

$c_0 =$ {SSN}
$c_1 =$ {Name,Illness}
$c_2 =$ {Name,Cause}
$c_3 =$ {DoB,ZIP,Illness}
$c_4 =$ {DoB,ZIP,Cause}
$c_5 =$ {Job,Illness}
$c_6 =$ {Job,Cause}

(b)

Fig. 1. An example of relation (a) and of confidentiality constraints over it (b)

Example 1. Figure 1 illustrates plaintext relation PATIENT (a) and the confidentiality constraints defined over it (b).

- c_0 is a singleton constraint indicating that the list of SSNs of patients is considered sensitive.
- c_1 and c_2 state that the association of patients' names with their illnesses and with the causes of the illnesses, respectively, is considered sensitive.
- c_3 and c_4 state that the association of patients' dates of birth and ZIP codes with their illnesses and with the causes of the illnesses, respectively, is considered sensitive; these constraints derive from c_1 and c_2 and from the fact that DoB and ZIP together could be exploited to infer the name of patients (i.e., they can work as a quasi-identifier [10]).
- c_5 and c_6 state that the association of patients' jobs with their illnesses and causes of the illnesses, respectively, is considered sensitive, since it could be exploited to establish a correlation between the job and the illness of a patient.

The satisfaction of a constraint c_i clearly implies the satisfaction of any constraint c_j such that $c_i \subseteq c_j$. We are therefore interested in enforcing a set $\mathcal{C} = \{c_1, \ldots, c_m\}$ of *well defined* constraints, where $\forall c_i, c_j \in \mathcal{C}, i \neq j,\ c_i \not\subset c_j$.

3 Correct and Minimal Fragmentation

Given a set $\mathcal{C}$ of confidentiality constraints over relation R, our goal is then to split R into two fragments F_o and F_s, in such a way that all sensitive data and

associations are protected. F_o is stored at the client (owner) side, while F_s is stored at the external server side. It is easy to see that, since there is no encryption, singleton constraints can be protected only by storing the corresponding attributes at the client side only. Therefore, each singleton constraint $c=\{a\}$ is enforced by inserting a into F_o and by not allowing a to appear in the schema of F_s. Association constraints are enforced via fragmentation, that is, by splitting the attributes composing the constraints between F_o and F_s. The resulting fragmentation $\mathcal{F}=\langle F_o, F_s\rangle$ should then satisfy two important requirements: *1)* all attributes in R should appear in at least one fragment to avoid loss of information; *2)* the confidentiality constraints should be properly protected, meaning that from the fragment stored at the external server (F_s) it should not be possible to reconstruct the content of the original relation R. These two requirements are formally captured by the following definition of *correct fragmentation*.

Definition 2 (Fragmentation correctness). *Let $R(a_1, \ldots, a_n)$ be a relation schema, $\mathcal{C}=\{c_1, \ldots, c_m\}$ be a set of well defined confidentiality constraints over R, and $\mathcal{F}=\langle F_o,F_s\rangle$ be a fragmentation for R, where F_o is stored at the client and F_s is stored at a storage server.*
$\mathcal{F}$ is a correct fragmentation *for R, with respect to $\mathcal{C}$, iff: 1) $F_o \cup F_s = R$ (completeness) and 2) $\forall c \in \mathcal{C}$, $c \not\subseteq F_s$ (confidentiality).*

The first condition requires every attribute in the schema of the original relation R to be represented in at least a fragment. The second condition requires the fragment stored at the external storage server F_s to not be a superset of any constraint. Note that the second condition applies only to F_s since F_o, under the client control and therefore accessible only to authorized users, can contain sensitive data and associations. For instance, fragmentation $\mathcal{F}=\langle\{$`SSN`,`Name`,`ZIP`,`Job`$\}$, $\{$`DoB`,`Illness`,`Cause`$\}\rangle$ is a correct fragmentation for PATIENT in Fig. 1(a), with respect to the confidentiality constraints in Fig. 1(b).

Since the client is supposed to have a more expensive storage, we are interested in computing a correct fragmentation that *minimizes* the client's storage due to the direct management of F_o. We then require that the two fragments F_o and F_s be disjoint, as formally stated by the following definition of *non redundant* fragmentation.

Definition 3 (Non redundant fragmentation). *Let $R(a_1, \ldots, a_n)$ be a relation and $\mathcal{C}$ a set of well defined constraints over R. A fragmentation $\mathcal{F}=\langle F_o,F_s\rangle$ of R is* non redundant *iff $F_o \cap F_s = \emptyset$.*

The non redundancy property does not affect the correctness of a fragmentation (Definition 2). Suppose that $\mathcal{F}=\langle F_o,F_s\rangle$ is a correct fragmentation for R with respect to $\mathcal{C}$, and that $F_o \cap F_s \neq \emptyset$. Any attribute a that appears in both F_o and F_s can be removed from F_o without violating Definition 2, since a is still represented in F_s and does not violate any constraint. In other words, a correct fragmentation can always be made non redundant by removing from F_o the attributes belonging to the intersection between F_o and F_s. For instance, consider fragmentation $\mathcal{F}'=\langle\{$`SSN`,`Name`,`ZIP`,`Job`,`Illness`,`Cause`$\}$,

{DoB,Illness,Cause}⟩, which is a correct fragmentation for relation PATIENT in Fig. 1(a) with respect to the confidentiality constraints in Fig. 1(b). Fragmentation $\mathcal{F}$=⟨{SSN,Name,ZIP,Job}, {DoB,Illness,Cause}⟩ obtained by removing the attributes in $F_o \cap F_s$={Illness,Cause} from F_o is still correct and satisfies Definition 3.

Our problem then consists in computing a fragmentation that is correct and non redundant and that minimizes the storage at the client site. Let $size(a)$ be the physical size of attribute a and $size(\mathcal{F})$ be the size of the fragmentation $\mathcal{F}$, computed as the physical size of the attributes composing F_o, that is, $size(\mathcal{F})=\sum_{a \in F_o} size(a)$. Our problem can be formally defined as follows.

Problem 1 (Minimal storage). Given a relation $R(a_1, \ldots, a_n)$ and a set of well defined confidentiality constraints $\mathcal{C}=\{c_1, \ldots, c_m\}$ over R, compute a fragmentation $\mathcal{F}=\langle F_o, F_s \rangle$ that satisfies the following conditions:

1. $\mathcal{F}$ is correct (Definition 2);
2. $\mathcal{F}$ is non redundant (Definition 3);
3. $\nexists \mathcal{F}'$ such that $size(\mathcal{F}') < size(\mathcal{F})$ and $\mathcal{F}'$ satisfies the two conditions above.

Note that, given a relation R and a set of confidentiality constraints $\mathcal{C}$ over it, there may exist different minimal storage fragmentations, all characterized by the same size (i.e., with the same size for F_o, which may however be composed of a different subset of attributes of R). We consider all such solutions equivalent.

Note that, whenever the physical size of the attributes composing R is not known (or the size of the attributes in R is similar and therefore cannot be considered as a discriminating factor for choosing attributes to be stored at the client site), Problem 1 can be exploited to compute a fragmentation that minimizes the number of attributes stored at the client site by simply setting $size(a)$=1, for all $a \in R$.

Example 2. Consider relation PATIENT and the confidentiality constraints on it in Fig. 1 and suppose that: $size$(SSN)=9, $size$(Name)=15, $size$(DoB)=8, $size$(ZIP)=5, $size$(Job)=10, $size$(Illness)=15, and $size$(Cause)=100. Fragmentation $\mathcal{F}$ = ⟨{SSN,Name,ZIP,Job}, {DoB,Illness,Cause}⟩, with $size(\mathcal{F})$= 9+15+ 5+10 = 39, is a minimal storage fragmentation for relation PATIENT with respect to $\mathcal{C}$. Suppose now that for all attributes a in PATIENT, $size(a)$=1. Fragmentation $\mathcal{F}'$ = ⟨{SSN,Illness,Cause}, {Name,DoB,ZIP,Job}⟩ is a minimal (storage) fragmentation, minimizing the number of attributes composing F_o.

In the following, we formally analyze the minimal storage problem, proving that it is NP-hard, and we present how a well-known approximation algorithm can be adapted for its solution.

4 Analysis of the Minimization Problem

The minimal storage problem (Problem 1) directly corresponds to the classical NP-hard *Weighted Minimum Hitting Set* problem [13], which can be formulated

INPUT
$R(a_1, \ldots, a_n)$ /* relation schema */
$\mathcal{C}=\{c_1, \ldots, c_m\}$ /* well-defined constraints */
$size(a)$, for all $a \in R$ /* physical size of attributes */

OUTPUT
A correct, non-redundant fragmentation $\mathcal{F}=\langle F_o, F_s \rangle$

MAIN
$to\_solve := \mathcal{C}$ /* constraints to be solved */
$F_o := \emptyset$ /* current solution */
while $to\_solve \neq \emptyset$ **do**
  Let $a \in (R \setminus F_o)$ be the attribute maximizing $|\{c \in to\_solve: a \in c\}|/size(a)$
  $to\_solve := to\_solve \setminus \{c \in to\_solve: a \in c\}$
  $F_o := F_o \cup \{a\}$
$F_s := R \setminus F_o$
return$(\langle F_o, F_s \rangle)$

Fig. 2. Approximation algorithm for the minimal storage problem

as follows: *Given a finite set S, a collection C of subsets of S, a weight function $w : S \to \mathbb{R}^+$, find a hitting set S', that is, a subset S' of S containing at least one element for each subset in C, such that $w(S') = \sum_{s \in S'} w(s)$ is minimum.*

The correspondence follows from the observation that any solution of the minimal storage problem must insert in F_o at least one attribute from each constraint in $\mathcal{C}$ to guarantee fragmentation correctness (Definition 2). The minimal storage problem can then be formulated as the problem of finding the set of attributes with lowest size (to be inserted in F_o) for breaking each constraint. It is then immediate to see that there is a correspondence between the two problems (minimal storage and weighted minimum hitting set) that can be determined by taking R as the finite set S, $\mathcal{C}$ as the collection C, and by taking as a weight function w the attribute size (i.e., $w(a) = size(a)$). Since the two problems are equivalent, the minimal storage problem (Problem 1) is NP-hard.

We also note that, by fixing $size(a)=1$ for all $a \in R$, the minimal storage problem directly corresponds to the classical NP-hard *Minimum Hitting Set* problem [13], which can be formulated as follows: *Given finite set S and a collection C of subsets of S, find a hitting set S', that is, a subset S' of S containing at least one element for each subset in C, such that $|S'|$ is minimum.* The minimum hitting set problem is equivalent to the minimum set covering problem [3]. Therefore, approximation algorithms and nonapproximability results for the minimum set covering problem also apply to the minimum hitting set problem. More precisely, the two problems are approximable within $(1 + \ln |S|)$ in polynomial time [16]. It is interesting to note that this approximation ratio is particularly accurate, since the minimum set cover, as well as the minimum hitting set, is not approximable within $(1 - \varepsilon) \ln |S|$, for any $\varepsilon > 0$ unless NP $\subset$ DTIME$(n^{\log \log n})$ [12]. These results on the approximability of the

iteration	*to_solve*	$\lvert\{c \in to\_solve: a \in c\}\rvert / size(a)$							F_o
		SSN	Name	DoB	ZIP	Job	Illness	Cause	
1	$c_0,c_1,c_2,c_3,c_4,c_5,c_6$	1/9	2/15	2/8	2/5	2/10	3/15	3/100	ZIP
2	c_0,c_1,c_2,c_5,c_6	1/9	2/15	0	–	2/10	2/15	2/100	ZIP,Job
3	c_0,c_1,c_2	1/9	2/15	0	–	–	1/15	1/100	ZIP,Job,Name
4	c_0	1/9	–	0	–	–	0	0	ZIP,Job,Name,SSN
	$\emptyset$	–	–	0	–	–	0	0	ZIP,Job,Name,SSN

Fig. 3. An example of execution of the algorithm in Fig. 2

minimum hitting set problem also apply to the weighted hitting set problem and, consequently, to the minimal storage problem (Problem 1).

The classical approximation algorithm for the weighted minimum hitting set problem [16] (with approximation ratio $(1 + \ln |S|)$) follows a greedy strategy. Figure 2 represents this approximation algorithm working on an instance of our minimal storage problem (Problem 1). Initially, the set *to_solve* of constraints to be solved is initialized to $\mathcal{C}$ and F_o is initialized to the empty set. At each iteration of the **while** loop, the algorithm chooses the attribute a that does not belong to F_o and that maximizes the ratio between the number of constraints in *to_solve* in which it is involved, and a's size. (The non-weighted version of the algorithm simply chooses the attribute that maximizes the number of non solved constraints in which it is involved.) Hence, a is inserted into F_o and set *to_solve* of non solved constraints is updated removing the constraints in which the chosen attribute a is involved. The **while** loop terminates when all constraints are solved (*to_solve*=$\emptyset$). Finally, F_s is obtained as the complement of F_o with respect to R.

Example 3. Figure 3 represents the execution, step by step, of the algorithm in Fig. 2 on relation PATIENT in Fig. 1(a), considering the confidentiality constraints in Fig. 1(b), and supposing *size*(SSN)=9, *size*(Name)=15, *size*(DoB)=8, *size*(ZIP)=5, *size*(Job)=10, *size*(Illness)=15, and *size*(Cause)=100. The first column in the table represents the set of constraints that are still unsolved; the subsequent seven columns represent, for each attribute in R, the number of unsolved constraints in which they are involved, divided by the attribute's size; the last column represents the attributes composing F_o. Symbol – associated with an attribute means that the attribute already belongs to F_o and therefore it does not need to be considered anymore. The solution computed by the algorithm is $\mathcal{F} = \langle\{$SSN,Name,ZIP,Job$\}, \{$DoB,Illness,Cause$\}\rangle$, which is a minimal fragmentation for the considered example.

5 Query Execution

The fragmentation of a relation R implies that only the two fragments F_o and F_s, which are stored in place of the original relation to satisfy confidentiality

F_o

tid	SSN	Name	ZIP	Job
1	123-45-6789	Alice	20051	Secretary
2	987-65-4321	Bob	22034	Student
3	147-85-2369	Carol	22039	Secretary
4	963-85-2741	David	20051	Lawyer
5	789-65-4123	Emma	22035	Student
6	123-65-4789	Fred	22034	Accountant

F_s

tid	DoB	Illness	Cause
1	80/02/11	asthma	Dust allergy
2	85/05/22	fracture	Car accident
3	73/07/30	carpal tunnel	Secretary
4	75/11/26	hypertension	Stress
5	90/03/15	asthma	Student
6	68/08/07	hypertension	Wrong diet

Fig. 4. Physical fragments corresponding to the fragmentation computed by the approximation algorithm on the relation and confidentiality constraints in Fig. 1

constraints, are used for query execution. However, since users authorized to access the original relation R should not worry about whether or not R has been fragmented, they formulate queries referring to R. Such queries need then to be translated into equivalent queries operating on F_o and/or F_s.

To guarantee the reconstruction of the content of the original relation R (lossless join property), at the physical level the two fragments F_o and F_s must have a common key attribute [2]. We then assume F_o and F_s have a common tuple id (attribute `tid`) that can be either: *1)* the key attribute of the original relation, if it is not sensitive, or *2)* an attribute that does not belong to the schema of the original relation R and that is added to F_o and F_s during the fragmentation process. Figure 4 illustrates the physical fragments for the minimal storage fragmentation of Example 3.

We consider SELECT-FROM-WHERE SQL queries of the form q="SELECT A FROM R WHERE C", where A is a subset of the attributes in R, and C=$\bigwedge_i cnd_i$ is a conjunction of basic conditions of the form $(a_i\ op\ v)$, $(a_i\ op\ a_j)$, or $(a_i$ IN $\{v_1, \ldots, v_k\})$, with a_i and a_j attributes in R, $\{v, v_1, \ldots, v_k\}$ constant values in the domain of a_i, and op a comparison operator in $\{=, >, <, \leq, \geq, \neq\}$. In the following, $Attr(cnd_i)$ is used to denote the attributes on which condition cnd_i operates. We now describe the query translation process.

5.1 Classification of Conditions

Condition $C = \bigwedge_i cnd_i$ in the WHERE clause of a query can be split in three conditions, namely C_o, C_s, and C_{so}, depending on the attributes involved in

the subexpression and that determine the party (client and/or storage server) responsible for its evaluation.

- $C_o = \bigwedge_i cnd_i : Attr(cnd_i) \subseteq F_o$ is the conjunction of the conditions that can be evaluated only by the *client*, independently from the server, since they involve only attributes stored at the client.
- $C_s = \bigwedge_i cnd_i : Attr(cnd_i) \subseteq F_s$ is the conjunction of the conditions in C that can be evaluated by the *storage server*, independently from the client, since they involve only attributes stored at the remote server. Note that, since the attributes stored at the server can be safely communicated to the client, C_s could also be evaluated by the client. However, this last option is highly impractical and should be avoided, because it requires to send to the client the projection over F_s of the attributes $Attr(cnd_i)$, for each cnd_i in C_s. This option would reduce the advantages of data outsourcing.
- $C_{so} = \bigwedge_i cnd_i : Attr(cnd_i) \cap F_s \neq \emptyset \wedge Attr(cnd_i) \cap F_s \neq \emptyset$ is the conjunction of conditions in C of the form $(a_i \ op \ a_j)$, where $a_i \in F_o$ and $a_j \in F_s$ or viceversa. Therefore, these conditions require information from the server and from the client to be evaluated, since they involve both attributes stored at the client and attributes stored at the server. Note that the conditions in C_{so} involve attributes of F_o that cannot be released to the external server, since they would possibly violate confidentiality constraints. Therefore, C_{so} can be evaluated only by the client.

Example 4. Consider the fragmentation in Fig. 4 and the query retrieving the names and dates of birth of people affected by asthma, living in area 22034 or 20051, and such that the cause of their illness is their job:

SELECT `Name`, `DoB`
FROM `Patient`
WHERE (`ZIP` IN $\{22034, 20051\}$) AND (`Illness`="asthma") AND (`Cause`=`Job`)

C_o = {`ZIP` IN $\{22034, 20051\}$}, since attribute `ZIP` belongs to F_o; C_s = {`Illness` = "asthma"}, since attribute `Illness` belongs to F_s; and, C_{so} = {`Cause` = `Job`}, since attributes `Job`$\in F_o$ and `Cause`$\in F_s$.

5.2 Query Evaluation Strategies

The evaluation process of a query q on R can follow two different strategies, depending on the order in which conditions C_o, C_s, and C_{so} are evaluated. The *Server-Client* strategy evaluates C_s at the server side and then evaluates both C_o and C_{so} at the client side. The *Client-Server* strategy first evaluates C_o at the client side, evaluates C_s at the server side, and finally checks C_{so} at the client side again. The left-hand side of Fig. 5 illustrates the two algorithms implementing the Server-Client and Client-Server strategies, respectively, executed by the client for translating and evaluating q. In the following, we briefly illustrate the working of the two algorithms.

Algorithm	Example
Server-Client strategy	
1. Let q = SELECT A FROM R WHERE C	q = SELECT Name, DoB FROM Patient WHERE (ZIP IN {22034,20051}) AND (Illness = "asthma") AND (Cause = Job)
2. Split C into subexpressions C_o, C_s, and C_{so}	C_o={ZIP IN{22034,20051}}; C_s={Illness="asthma"}; C_{so}={Cause=Job}
3. A_{qs}:=$(F_s \cap A) \cup \{a \in F_s \mid \exists cnd \in C_{so}, a \in Attr(cnd)\}$	A_{qs} = {DoB, Cause}
4. Send q_s = SELECT tid, A_{qs} FROM F_s WHERE C_s to the storage server	q_s = SELECT tid,DoB,Cause FROM F_s WHERE Illness = "asthma"
5. Receive the result R_s of q_s from the server	
6. Execute q_{so} = SELECT A FROM F_o JOIN R_s ON F_o.tid=R_s.tid WHERE $C_o \wedge C_{so}$	q_{so} = SELECT Name, DoB FROM F_o JOIN R_s ON F_o.tid=R_s.tid WHERE (ZIP IN {22034,20051}) AND (Cause = Job)
7. Return the result R_q of q_{so} to the user	
Client-Server strategy	
1. Let q = SELECT A FROM R WHERE C	q = SELECT Name, DoB FROM Patient WHERE (ZIP IN {22034,20051}) AND (Illness = "asthma") AND (Cause = Job)
2. Split C into subexpressions C_o, C_s, and C_{so}	C_o={ZIP IN{22034,20051}}; C_s={Illness="asthma"}; C_{so}={Cause=Job}
3. A_{qs}:=$(F_s \cap A) \cup \{a \in F_s \mid \exists cnd \in C_{so}, a \in Attr(cnd)\}$	A_{qs} = {DoB, Cause}
4. Execute q_o = SELECT tid FROM F_o WHERE C_o	q_o = SELECT tid FROM F_o WHERE ZIP IN {22034,20051}
5. Send the result R_o of q_o to the storage server	
6. Send q_s = SELECT tid, A_{qs} FROM F_s JOIN R_o ON F_s.tid=R_o.tid WHERE C_s to the storage server	q_s = SELECT tid,DoB,Cause FROM F_s JOIN R_o ON F_s.tid=R_o.tid WHERE Illness = "asthma"
7. Receive the result R_s of q_s from the server	
8. Execute q_{so} = SELECT A FROM F_o JOIN R_s ON F_o.tid=R_s.tid WHERE C_{so}	q_{so} = SELECT Name, DoB FROM F_o JOIN R_s ON F_o.tid=R_s.tid WHERE Cause = Job
9. Return the result R_q of q_{so} to the user	

Fig. 5. Algorithm for evaluating query q on fragmentation $\mathcal{F}$ and an example of its execution

– *Server-Client* strategy. The basic idea is that first the server evaluates the conditions in C_s on F_s, and then the client refines the result by filtering out the tuples that do not satisfy either C_o or C_{so}.

The corresponding algorithm receives in input the query q (step 1) to be evaluated and returns the relation R_q resulting from its evaluation. The algorithm then splits the condition C in the WHERE clause in three subexpressions, C_o, C_s, and C_{so}, on the basis of the attributes involved in the basic conditions composing C (step 2). The algorithm identifies the set A_{q_s} of attributes that belong to F_s, but that are necessary to the client for completing the evaluation of q, since either they belong to A or appear in a basic

condition in C_{so} (step 3). The algorithm then defines, and sends to the storage server, the query q_s operating on F_s and evaluating condition C_s (step 4). The SELECT clause of this query is composed of A_{q_s} and attribute `tid`, which is necessary to join the result of q_s with F_o. Then the server computes and returns the result R_s of the evaluation of q_s to the client (step 5). The client defines and directly executes the query q_{so} operating on the join between R_s and F_o and evaluating both C_o and C_{so} (step 6). The relation resulting from the evaluation of query q_{so} corresponds to the result of the original query q, which is then returned to the user (step 7).

– *Client-Server* strategy. The basic idea is that first the client evaluates the conditions in C_o on F_o, the server then refines the result by filtering out the tuples that do not satisfy C_s, and finally the client discards the tuples that do not satisfy C_{so}. The first three steps of the corresponding algorithm are the same as for strategy Server-Client. After having split the conditions (step 2) and identified attributes A_{q_s} (step 3), the client defines and executes the query q_o operating on F_o and evaluating condition C_o (step 4). The SELECT clause of this query is composed of attribute `tid` only, which is the only attribute that can be communicated to the remote server (step 5) and is needed to perform join. The algorithm sends to the storage server the query q_s, operating on the join between R_o and F_s and evaluating condition C_s (step 6) for execution. The SELECT clause of query q_s (as for Server-Client strategy) is composed of A_{q_s} and attribute `tid`. The server computes q_s and returns its result R_s to the client (step 7). The client defines and directly executes query q_{so} operating on the join between R_s and F_o and evaluating C_{so} (step 8). The relation resulting from the evaluation of query q_{so}, corresponding to the result of the original query q, is returned to the requesting user (step 8).

Example 5. Consider relation PATIENT in Fig. 1(a), its fragmentation in Fig. 4, and the query q introduced in Example 4. $A_{q_s} = \{$`DoB`,`Cause`$\}$, since `DoB` belongs to the SELECT clause of the original query, while `Cause` is involved in a condition in C_{so}. The right-hand side of Fig. 5 illustrates the queries generated by the algorithm for translating q in a set of equivalent queries operating on $\mathcal{F}$, with the Server-Client and Client-Server strategies.

The choice between the Server-Client and Client-Server strategies depends on the possible leakage of information that the Client-Server strategy may cause. If the storage server is supposed to know or can infer query q (e.g., because the query is publicly available) the Client-Server strategy cannot be adopted because the storage server can infer that the tuples in R_o are all and only the tuples that satisfy C_o, thus causing a leakage of information. As an example, consider query q = "SELECT `Illness` FROM `Patient` WHERE `Name` = "Alice"" over relation PATIENT. If we adopt the Client-Server strategy, q_o = "SELECT `tid` FROM F_o WHERE `Name` = "Alice"" returns only one tuple with `tid`=1. Knowing q, the storage server can, from the result, reconstruct the sensitive associations between `Name` and `Illness` and between `Name` and `Cause` (violating c_1 and c_2)

for the tuple with `tid`=1. To avoid information leakage, the client can add noise to the result of query q_o, by artificially inserting the id of some tuples that do not satisfy C_o. In this case, query q_{so} should evaluate both C_{so} and C_o, to remove from the final result the tuples artificially added to R_o.

If the storage server does not to know (and cannot infer) q or R_o is adequately protected, both the Server-Client and Client-Server strategies can be adopted without privacy violations. In this case, the choice between the two strategies can only be based on performances. Following the criterion usually adopted by distributed database systems, the most selective condition is evaluated first (i.e., the sub-query with the smallest result is anticipated). Therefore, if C_o is more selective than C_s, we adopt the Client-Server strategy and viceversa, if C_s is more selective than C_o, we adopt the Server-Client strategy. Note that the selectivity of query q_o needs to consider also the possible noise added to R_o for privacy purposes.

6 Related Work

Most of the research on the outsourced data paradigm assume the data to be entirely encrypted, focusing on the design of techniques for the efficient execution of queries (Database As a Service paradigm) [7,14,15,19]. The first proposal suggesting the combined use of fragmentation and encryption for enforcing privacy constraints has been presented in [1]. This technique is based in the assumption that data are split over two non communicating honest-but-curious database servers and resorts to encryption any time two fragments are not sufficient for enforcing confidentiality constraints. The model presented in [8,9] removes the limiting assumption that the two storage servers must not communicate, by splitting the data over different fragments. The advantage is that only sensitive attributes need to be encrypted, while sensitive associations can always be solved via fragmentation.

In this paper, differently from previous approaches, we aim at solving confidentiality constraints without resorting to encryption, by storing a portion of the sensitive data at the client site. The main advantage of this novel solution is that it simplifies the architecture and produces a system characterized by better maintenance, avoiding the issues related with key management.

On another line of related work, classical proposals on the management of queries in centralized and distributed systems [4,6,17] describe how efficient query plans can be obtained. These solutions, however, cannot be applied in our context, since they do not take into account possible information leakage to the storage server, which can violate confidentiality constraints.

An affinity to the work presented in this paper can be found in [5,11]. Although these approaches share with our problem the common goal of enforcing confidentiality constraints on data, they are concerned with retrieving a data classification (according to a multilevel mandatory policy) that ensures sensitive information is not disclosed and do not consider the fragmentation technique.

The problem of fragmenting relational databases has been addressed in the literature [18], with the main goal of improving query evaluation efficiency.

However, these approaches are not applicable to the considered scenario, since they do not take into consideration privacy requirements.

7 Conclusions

The paper presented an approach for the management of relational data where a honest-but-curious server is used to manage all the data that do not permit to violate confidentiality constraints. Given the continuous increase in the size, variety, and accessibility needs of data collections containing sensitive information, together with the availability of novel network-enabled computing platforms, we envision a significant domain for the application of the techniques presented here or their extensions. We see an opportunity for extending this work to more general databases and network applications, like Web-based management of email and office automation tasks, remote backup and data outsourcing services, and rentable computational services.

Acknowledgements

This work was supported in part by the EU within the 7FP project under grant agreement 216483 "PrimeLife". The work of Sushil Jajodia was partially supported by the National Science Foundation under grants CT-0716323, CT-0627493, and IIS-04300402 and by the Air Force Office of Scientific Research under grants FA9550-07-1-0527 and FA9550-08-1-0157. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsoring organizations.

References

1. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: A distributed architecture for secure database services. In: Proc. of the 2nd Biennial Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, CA, USA (January 2005)
2. Atzeni, P., Ceri, S., Paraboschi, S., Torlone, R.: Database systems - Concepts, languages and architectures. McGraw-Hill, New York (1999)
3. Ausiello, G., D'Atri, A., Protasi, M.: Structure preserving reductions among convex optimization problems. Journal of Computer and System Sciences 21(1), 136–153 (1980)
4. Bernstein, P., Goodman, N., Wong, E., Reeve, C., Rothnie, J.J.B.: Query processing in a system for distributed databases (SDD-1). ACM Transactions on Database Systems 6(4), 602–625 (1981)
5. Biskup, J., Embley, D., Lochner, J.: Reducing inference control to access control for normalized database schemas. Information Processing Letters 106(1), 8–12 (2008)
6. Ceri, S., Pelagatti, G.: Distributed Databases: Principles and Systems. McGraw-Hill, New York (1984)

7. Ceselli, A., Damiani, E., De Capitani di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Modeling and assessing inference exposure in encrypted databases. ACM Transactions on Information and System Security 8(1), 119–152 (2005)
8. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragmentation and encryption to enforce privacy in data storage. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 171–186. Springer, Heidelberg (2007)
9. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragmentation design for efficient query execution over sensitive distributed databases. In: Proc. of the 29th International Conference on Distributed Computing Systems (ICDCS 2009), Montreal, Canada (June 2009)
10. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: k-Anonymity. In: Yu, T., Jajodia, S. (eds.) Secure Data Management in Decentralized Systems. Springer, Heidelberg (2007)
11. Dawson, S., De Capitani di Vimercati, S., Lincoln, P., Samarati, P.: Maximizing sharing of protected information. Journal of Computer and System Sciences 64(3), 496–541 (2002)
12. Feige, U.: A threshold of $\ln n$ for approximating set cover. Journal of the ACM 45(4), 634–652 (1998)
13. Garey, M., Johnson, D.: Computers and intractability; a guide to the theory of NP-completeness. W.H. Freeman, New York (1979)
14. Hacigümüs, H., Iyer, B., Mehrotra, S.: Providing database as a service. In: Proc. of the 18th International Conference on Data Engineering (ICDE 2002), San Jose, CA, USA (February/March 2002)
15. Hacigümüs, H., Iyer, B., Mehrotra, S., Li, C.: Executing SQL over encrypted data in the database-service-provider model. In: Proc. of the 21st ACM SIGMOD International Conference on Management of Data, Madison, WI, USA (June 2002)
16. Johnson, D.: Approximation algorithms for combinatorial problems. In: Proc. of the ACM Symposium on Theory of Computing (STOC 1973), Austin, TX, USA (April/May 1973)
17. Kossmann, D.: The state of the art in distributed query processing. ACM Computing Surveys 32(4), 422–469 (2000)
18. Navathe, S., Ra, M.: Vertical partitioning for database design: A graphical algorithm. In: Proc. of the ACM SIGMOD International Conference on Management of Data, Portland, OR, USA (June 1989)
19. Wang, H., Lakshmanan, L.V.S.: Efficient secure query evaluation over encrypted XML databases. In: Proc. of the 32nd International Conference on Very Large Data Bases (VLDB 2006), Seoul, Korea (September 2006)

Data Is Key: Introducing the Data-Based Access Control Paradigm

Wolter Pieters and Qiang Tang

Faulty of Electrical Engineering, Mathematics and Computer Science
University of Twente
P.O. Box 217, 7500 AE ENSCHEDE, The Netherlands
{w.pieters,q.tang}@utwente.nl

Abstract. According to the Jericho forum, the trend in information security is moving the security perimeter as close to the data as possible. In this context, we suggest the idea of data-based access control, where decryption of data is made possible by knowing enough of the data. Trust is thus based on what someone already knows. A specific problem is defined as follows: given n pieces of data, an agent is able to recover all n items once she knows k of them. The problem is similar to both secure sketches and secret sharing, and we show that both can be used as a basis for constructions. Examples of possible applications are granting access without credentials, recovering forgotten passwords and sharing personal data in social networks.

Keywords: data-based access control, de-perimeterisation, secure sketches, secret sharing.

1 Introducing Data-Based Access Control

1.1 Motivation

According to the Jericho forum [4], traditional boundaries in information security are disappearing. Organisations are flexible and outsource part of their processes, employees work from home and mobile devices contain loads of data. Instead of relying on firewalls, the protection should therefore lie as close to the data as possible. In this paper, we address this issue from the perspective of access control.

In general, mainstream access control relies on two separate mechanisms: authentication and authorisation. It is first determined if the user has the claimed identity, and based on this identity certain privileges or trust are assigned. In a de-perimeterised setting, however, we may not be so sure anymore about the identity of persons, or the current privileges that should be coupled to it. Because people join and leave organisations on a regular basis, information about their identity may well be outdated. Especially when roles of persons change often and are not administered timely, identity can be an unreliable intermediary. In this context, the question can be asked whether the two steps can be merged into a single one. Can we trust people without knowing their identity?

Often, authentication and authorisation of persons is based on what a person knows. Normally, this is a username-password combination or a key. We radicalise this idea here, and focus on whether it is possible to let *any* piece of data serve as part of the

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 240–251, 2009.

access control policy. More specifically, we concentrate on access control policies in which the data protects itself. That is, an agent acquires access to new data based on the data it already possesses. We call this approach *data-based access control*. In the data-based access control approach, identity is no longer the focal point of authorisations. Instead, authorisations are coupled directly to credentials. These credentials can be any form of data: passwords, keys, files, or other pieces of data.

As an example, consider an electronic health record of a patient. We assume here that the file resides in a protected form on a public server. Either the authorities or the patient herself will have to administer the access policies for the file. How can they set the access control such that a practitioner can update her copy of the file to a new version, without first having to authenticate herself? Most importantly, how can we assign the access rights for the current file without relying on authentication? Basically, practitioners that have been treating the patient before will possess an earlier version of the medical file, including part of the information of the new version. This means that the patient can grant access to the updated version based on the earlier version as a credential. Apart from practical challenges in managing such access rights, the key feature is that these rights are based on data rather than special credentials.

If the patient does not trust the possessors of the previous version anymore, she may set a different access policy for the new file, in which an additional credential is required. She may then distribute this credential to the trusted practitioners via a different channel. How such policies can be implemented will be discussed in the following. The main point is that it is possible to base access control on information rather than identity.

In this paper, we define the characteristics of data-based access control and the associated policies as a new framework for data protection. We also show that existing cryptographic primitives can be used to implement the policies. The main contribution of this paper is therefore the definition of a new research area and its connection with already existing cryptographic primitives.

1.2 Problem Statement

Data-based access control is a form of access control where the possibility of decryption is protected by the data itself, not by separate credentials such as keys or attributes. Data-based access control is based on *similarity* between what the user wants to know and what she already knows. The central assumption is that if one already knows much within a specific domain, one has the right to know more.

Definition 1. Data-based access control *is the granting of access to information based on the similarity between the information being accessed and the information provided to access it.*

In this paper, we introduce (k, n) threshold data-based access control. More precisely, if from a set of n data-items the agent already knows k, she can recover the remaining items. Also, when the agent knows fewer than k, the uncertainty about the remaining items is equal to or only negligibly smaller than the entropy in each data item.

1.3 Applications

Data-based access control may be used for various purposes:

- authorisation without identification, where trust is based on what an agent already knows;
- recovering forgotten passwords; if an agent uses n passwords, she may store these in such a way that she can recover one if she remembers the other $n - 1$;
- sharing data in social networks; people may want to reveal more to people who already know a lot about them;
- secure version management; access can be granted to a new version of a file based on knowledge of earlier versions.

1.4 Related Work

Developments in the direction of data-level security include sticky policies [6] and attribute-based encryption [8]. The sticky policies paradigm involves the idea that policies can be associated to data in such a way that they are enforced by the association mechanism itself. Attribute-based encryption is such a mechanism.

In *attribute-based encryption* [8], the possibility of decryption is dependent on the receiver's attributes rather than her knowledge. These attributes have been verified by a key authority, which has issued a key corresponding to these attributes. In data-based access control, there is no independent key authority, since possession of part of the data immediately proves that one is entitled to know the rest.

An idea similar to data-based access control is the *fuzzy vault* [5]. There, a secret value is "locked" by a set of public values. If a sufficiently close set of public values is input, the vault can be unlocked. In a fuzzy vault, however, the elements for unlocking the vault are from a publicly announced domain, which is different from the secret we wish to protect. This is not the case in data-based access control: here, the elements for unlocking are themselves also the secrets.

Data-based access control can be conceived as a *secure sketch* problem. A secure sketch [1] is a procedure that maps a secret value to a public bit string, where the public bit string plus an approximation of the secret value allow for recovery of the secret value, provided the approximation is close enough. Both problems involve the publication of information that allows one to reconstruct the original data if one can approximate the original data closely enough. Indeed, the information published in data-based access control can be thought of as a secure sketch, namely one in which the distance measure is the set difference between the stored data-items and the remembered data-items. However, secure sketches have been mainly applied to problems where the input data is inherently fuzzy (like biometrics), where characteristics of the biometric template can be described in terms of feature sets. In this paper, we focus on the (k, n) threshold data recovery problem instead, where it may or may not be allowed to approximate some or all of the individual items. Also, as opposed to secure sketches, data-based access control is not focused on error correction. For decryption, an agent should input exactly k known values, not n values of which at least k are correct.

An approach similar to data-based access control was suggested for authentication purposes [3]. In this "personal entropy" scheme, an authentication secret is divided using Shamir's secret sharing [9]. For each share, a question together with the correct answer allows one to decrypt the share of the secret. The scheme that is proposed may also be applied in data-based access control. In the current work, however, the essential feature is that it is the data itself that is being protected, not an authentication key.

Recently, it was proposed to use digital objects as passwords [7]. The objects used in that framework are public information such as music or pictures rather than protected information. Also, the approach is again focused on acquiring a key rather than accessing the protected data directly.

1.5 Outline of the Paper

In section 2, we introduce the principles of data-based access control. In section 3, we describe a possible solution for the problem of (k, n) threshold data-based access control based on polynomial interpolation. In section 4, we show how an order-invariant scheme can be devised using secure sketches. In section 5, we discuss our prototype system. In section 6, we provide some suggestions for asymmetric schemes. In section 7, we evaluate the approach and solution. In section 8, we conclude and suggest further research.

2 Principles

In the traditional approach of access control, there are two phases: authentication and authorisation. Authentication specifies how the possession of certain credentials establishes an identity. Authorisation specifies how a certain identity leads to access.

Data-based access control does not distinguish a priori between credentials and data. It assumes a distribution over the participating agents of data-items from the set that we are interested in ($\mathcal{U}$) and defines how access to these items is related to access to other items. In the notation, we use $\mathcal{D}$ for sets of data items and D for individual data items.

We can distinguish between non-interactive and interactive forms of data-based access control. In interactive forms, agents can execute a protocol, which in the end provides the desired distribution of data (figure 1). In non-interactive forms, an agent publishes its data on a public bulletin board with the policies "attached", after which other agents with the appropriate possessions can access it. We focus on non-interactive forms here.

A data-based access control policy consists of a domain $\mathcal{U}$ and a function $\mathcal{P}(\mathcal{U}) \rightarrow \mathcal{P}(\mathcal{U})$, mapping sets of required items to sets of items that the required items give access to ($\mathcal{P}$ denotes power set). A policy is well-formed if a) data-items give access to themselves, and b) more data-items give access to more data-items.

Definition 2. *A data-based access control policy is* well-formed *if for each mapping* $\mathcal{D}_1 \mapsto \mathcal{D}_2$*:*

- $\mathcal{D}_1 \subseteq \mathcal{D}_2$*, and*
- *for each data item* D *and mapping* $\mathcal{D}_1 \cup \{D\} \mapsto \mathcal{D}_3$*:* $\mathcal{D}_2 \subseteq \mathcal{D}_3$.

The intuition here is that one already has access to the data-items one supplies as credentials, so a policy denying this access would not be enforceable. Moreover, if a set of data-items does not give one access to a specific piece of data, but a subset does, one can use the subset instead and still get the piece of data. Thus, again, such a policy would not be enforceable.

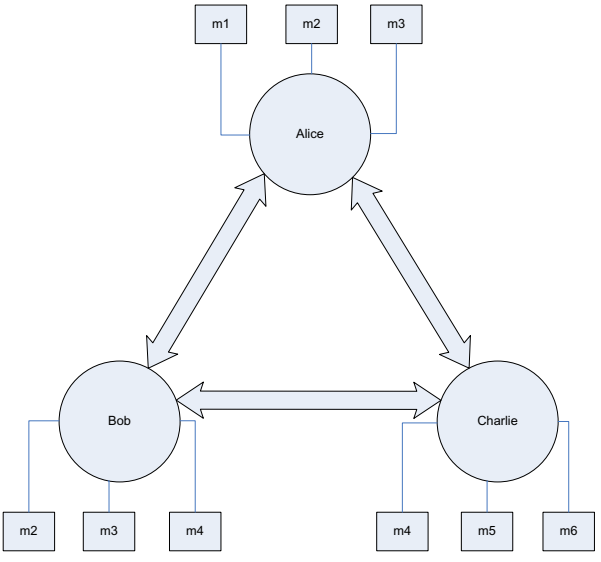


Fig. 1. Interactive data-based access control. Alice, Bob and Charlie each possess a set of messages. They wish to share all of their messages with another person if they already share at least two.

Example 1. We give an example of a well-formed policy:

- $\emptyset \mapsto \emptyset$
- $\{D_1\} \mapsto \{D_1\}$
- $\{D_2\} \mapsto \{D_2\}$
- $\{D_3\} \mapsto \{D_3\}$
- $\{D_1, D_2\} \mapsto \{D_1, D_2, D_3\}$
- $\{D_1, D_3\} \mapsto \{D_1, D_2, D_3\}$
- $\{D_2, D_3\} \mapsto \{D_2, D_3\}$
- $\{D_1, D_2, D_3\} \mapsto \{D_1, D_2, D_3\}$

We show that the fourth line satisfies the requirements. Trivially, $\{D_3\}$ is a subset of itself (first requirement). Also, adding a data item gives the mappings $\{D_1, D_3\} \mapsto \{D_1, D_2, D_3\}$ and $\{D_2, D_3\} \mapsto \{D_2, D_3\}$. In both cases, $\{D_3\}$ is a subset of the image (second requirement). The requirements can similarly be verified for the other lines.

We distinguish between *symmetric* and *asymmetric* data-based access control. Symmetric access control can only be used with symmetric policies. A symmetric policy is a well-formed policy in which the items in the rules are equivalent: if I can get a by knowing b, then I can also get b by knowing a. More precisely, if I can get any *additional* item D based on a set of known items $\{D_1 ... D_n\}$, I can also retrieve any item from this set if I do not know that item, but I do know D.

Definition 3. *A data-based access control policy is* symmetric *if it is well-formed, and for each mapping $\mathcal{D}_1 \mapsto \mathcal{D}_2$, for which $D_1 \in \mathcal{D}_1$, $D_2 \in \mathcal{D}_2$ and $D_2 \notin \mathcal{D}_1$, there also exists a mapping $(\mathcal{D}_1 \setminus \{D_1\}) \cup \{D_2\} \mapsto \mathcal{D}_3$, where $D_1 \in \mathcal{D}_3$.*

Example 2. The policy of the previous example is not symmetric. With D_1 and D_2, one can get D_3 (fifth line), but when reversing D_1 and D_3, one cannot get D_1 with D_2 and D_3 (seventh line).

Symmetric data-based access control is the focus in the current work; some ideas on asymmetry are discussed in section 6. More specifically, we focus on (k, n) threshold data-based access control.

Definition 4. *A data-based access control policy for universe $\mathcal{U}$ is a* (k, n) threshold policy *if:*

- $|\mathcal{U}| = n$,
- *for each $\mathcal{D} \subseteq \mathcal{U}$ with $|\mathcal{D}| \geq k$, $\mathcal{D} \mapsto \mathcal{U}$, and*
- *for each $\mathcal{D} \subseteq \mathcal{U}$ with $|\mathcal{D}| < k$, $\mathcal{D} \mapsto \mathcal{D}$.*

Lemma 1. *A (k, n) threshold policy is symmetric.*

Proof. We first prove that a (k, n) threshold policy is well-formed. Because $\mathcal{D}$ either maps to $\mathcal{U}$ or to itself, the first requirement of well-formedness is satisfied, since $\mathcal{D}$ is both a subset of itself and of $\mathcal{U}$. Furthermore, in the first case, $|\mathcal{D}| \geq k$ and therefore $|\mathcal{D} \cup \{D\}| \geq k$. $\mathcal{D} \cup \{D\}$ will thus also map to $\mathcal{U}$, which is a subset of itself, proving the second requirement. In the second case, $\mathcal{D} \cup \{D\}$ will map to either $\mathcal{D} \cup \{D\}$ or $\mathcal{U}$, of both of which $\mathcal{D}$ is a subset.

To prove that a (k, n) threshold policy is also symmetric, we look at the policy rule $\mathcal{D}_1 \mapsto \mathcal{D}_2$ and distinguish again the cases $|\mathcal{D}_1| \geq k$ and $|\mathcal{D}_1| < k$. In the first case, $\mathcal{D}_2 = \mathcal{U}$. For any $D_1 \in \mathcal{D}_1$ and $D_2 \notin \mathcal{D}_1$, $|(\mathcal{D}_1 \setminus \{D_1\}) \cup \{D_2\}| = |\mathcal{D}_1| \geq k$. Therefore, the new set will also map to $\mathcal{U}$, of which D_1 is a member. In the second case, $\mathcal{D}_2 = \mathcal{D}_1$ and there will be no item D_2 satisfying the condition $D_2 \in \mathcal{D}_2$ and $D_2 \notin \mathcal{D}_1$. Symmetry is thus trivially achieved. □

A construction for (k, n) threshold data-based access control is defined by two operators:

- an encryption function Enc, taking as input n data-items $\mathcal{U} = \{D_1 ... D_n\}$, the value k and outputting a public string P;
- a decryption function Dec, which takes as input the public value P and k data-items $\mathcal{T} = \{T_1 ... T_k\}$, and yields $\mathcal{U}$ if and only if $\mathcal{T} \subseteq \mathcal{U}$.

(k, n) threshold data-based access control should satisfy the following security properties:

- knowledge of k items allows exact reconstruction of $\mathcal{U}$;
- knowledge of fewer than k items reveals no additional information (information theoretic security) or only a negligible amount of information (computational security) about any of the other individual items;
- applying Dec with a set of items that is partially correct should not give information about which items are correct.

We can distinguish between *direct* and *indirect* data-based access control. In the former, the remaining data-items can be recovered without any intermediate secret value that

protects the data. In the latter, knowledge of enough data-items allows for recovering a key for a conventional encryption algorithm that can be used to decrypt the full data-set. Because the latter relies on standard encryption techniques, the security will typically be computational rather than information theoretic.

The forms of data-based access control discussed here can be used in securing various types of databases. To enable this in practice, we need schemes that actually realise the desired properties. In the following, we will propose such schemes for (k, n) threshold data-based access control.

3 A Polynomial Interpolation Scheme

3.1 Shamir's Secret Sharing

Shamir [9] introduced a scheme for secret sharing based on polynomial curve fitting. Secret sharing involves dividing a secret D into n parts, of which k are necessary and sufficient to reveal the secret. The idea is to define a polynomial of degree $k-1$, using D as one of the coefficients and choosing the others randomly, and generate n points on the polynomial as the parts of the secret. Because the degree is $k-1$, k shares will suffice to reconstruct the coefficients and thus the secret.

In Shamir's scheme, the coefficients are chosen randomly, except for the coefficient that represents the secret. This makes it possible to limit the degree of the polynomial to k. However, in data-based access control, we have n data-items to fit into the polynomial. It is not possible to represent these as n points on a polynomial of degree $k-1$, because in general n points will not lie on a single polynomial of this degree.

3.2 Direct Scheme

We present two possible solutions. The first is to represent the data-items as n points uniquely defining a polynomial of degree $\leq n-1$. Assume that we have n data-items D_i $(1 \leq i \leq n)$ and an injective two-way function h mapping data-items to numbers below a prime p. We now define a polynomial f of degree $\leq n-1$ in $\mathbb{Z}_p$ by the n points $(i, h(D_i))$. If we then publish $n-k$ additional points on the polynomial, someone who knows k of the data-items can reconstruct the polynomial, the points and thereby the data-items.

We publish:

- n and p
- $n-k$ points $(j, f(j))$ on the polynomial, where $j > n$

Decryption can now be done by reconstructing the polynomial from the $n-k$ public points and k known points (n in total). One can then recalculate the remaining points, and by applying the inverse of h recover the data items. This requires that the data-items be encoded using a two-way function, which may only be possible in case of small data-items, such as passwords. If we wish to use a hash function to map the data-items to numbers, this approach is not feasible, because points cannot be converted back to data items. An indirect and more general approach, where an additional key is added, is defined below.

3.3 Indirect Scheme

Assume that we have n data-items D_i ($1 \leq i \leq n$) and a function h mapping data-items to numbers below a prime p. The function h may for example be a hash function, but this is not required for security. (The numbers must be uniformly distributed though, to prevent an adversary from acquiring information from correlations of data-items.) We choose a random key S from $\mathbb{Z}_p$. In general, $n + 1$ points are necessary and sufficient to define or reconstruct a polynomial of degree $\leq n$. We now define a polynomial f of degree $\leq n$ in $\mathbb{Z}_p$ by the $n + 1$ points $(0, S)$ and $(i, h(D_i))$.

Suppose that someone already knows $(i, h(D_i))$ for k of the data-items. In order to recover the full polynomial, including the key S, one needs $n + 1 - k$ additional points of the polynomial. Therefore, we publish:

- the data encrypted with key S
- n and p
- $n + 1 - k$ points $(j, f(j))$ on the polynomial, where $j > n$

The data may be accompanied by a plaintext description, allowing agents to find the data-set they want, as well as to specify the order of the items. Note that the value of k can be derived from n and the number of published points, and thus does not need to be published.

Since the $(n+1-k)$ points on the polynomial made public represent additional points on the polynomial, these will allow anyone who knows $(i, h(D_i))$ for k of the data-items to recover f from the $(n + 1 - k) + k = n + 1$ points in his possession. He can then recover the key S from the point $(0, S)$ and decrypt all the data. The additional points made public thus provide information about the relation between the data-items, without revealing additional information about the data-items themselves, as shown in [9].

Although it would be possible to use a polynomial of degree $n-1$ based on the points $(i, h(D_i))$ and then define $S \equiv f(0)$, this would mean that the key would depend on the data that the key is used to encrypt, leading to possible weaknesses in the encryption scheme employed in practice. We therefore choose to use a random key and polynomial of degree n.

3.4 Example

As an example, assume we have a data-set with the following description: Alice's passwords: (1, laptop), (2, mail), (3, social networking). We assume (without loss of generality) that the passwords are numbers. For explanation purposes, we use in the example the small values 24, 37 and 62 respectively. We also use $p = 67$ here. We choose as the secret key S the value 45. Now, we have the following points on the polynomial: $(0, 45)$, $(1, 24)$, $(2, 37)$, $(3, 62)$. Using a polynomial curve fitting algorithm, we can find that $f(x) = 41x^3 + 28x^2 + 44x + 45$. We wish to be able to recover one password as long as we remember the other two. Therefore, we publish two additional points on the polynomial: (4,10) and (5,60). This means that the following information is published:

- the description of the data-set: Alice's passwords: (1, laptop), (2, mail), (3, social networking)

- the data encrypted with key S (not shown here because of the simplicity of the example)
- $n = 3, p = 67$
- the additional points (4,10) and (5,60)

Now assume that Alice lost her password 3 (her social networking password). She remembers the other two passwords, though. She therefore possesses four points on the polynomial: $(1, 24)$, $(2, 37)$, $(4, 10)$ and $(5, 60)$. This is enough to obtain by polynomial curve fitting the polynomial $f(x) = 41x^3 + 28x^2 + 44x + 45$. Calculating $f(0) = 45$ gives Alice the key to decrypt all her passwords, including the missing one. Note that all computations can be done offline after obtaining the public data.

4 An Order-Invariant Scheme

For obtaining an order-invariant scheme, we apply the set difference construction of a secure sketch from [1]. For the direct scheme, assume again that we have an injective two-way function h mapping data-items to numbers below a prime p. Here, the secret values $h(D_i)$ are encoded as the x-coordinates for which the value of a polynomial f coincides with the value of a polynomial g. Since the values are encoded as x-coordinates as opposed to the y-coordinates in the previous scheme, their order is unimportant.

In [1], Reed-Solomon error correction is used to account for possibly incorrect set elements that are supplied in the decoding stage. Because we are not interested in error correction here, we can leave out this possibility. For decryption, one should input k values which are all correct. This allows us to simplify the scheme from [1] as follows.[1]

The encryption algorithm Enc now takes the following steps (again working in polynomials over $\mathbb{Z}_p$):

1. Choose a secret polynomial g of degree $k - 1$ at random;
2. Calculate and publish the unique monic polynomial f of degree exactly n by solving $f(h(D_i)) = g(h(D_i))$ for all D_i. Publish k as well.

Decryption now works as follows:

1. Reconstruct g by solving $f(h(D_i)) = g(h(D_i))$ for all known items;
2. Find the remaining values for which $f(h(x)) = g(h(x))$; these are the other data-items.

Since there is no error correction needed, there is minimal entropy loss. That is to say, given $k - 1$ values for which $f(x) = g(x)$, one learns nothing about the remaining root of $f(x) - g(x)$.

It is also possible to use this scheme for indirect data-based access control. In [1] it is shown that a secure sketch can be turned into a *fuzzy extractor*. A fuzzy extractor outputs a secure key based on the input of the secure sketch. When we turn the secure sketch, in our case (Enc, Dec), into a fuzzy extractor, we can thus obtain a key for indirect data-based access control. Alternatively, an extra data-item may be added that represents the key for decrypting the data (as we did in the ordered scheme).

[1] In a later version of their publication [2], the authors use a deterministic rather than a probabilistic variant of their algorithm. The following algorithms can be adapted accordingly.

5 Implementation

A prototype of the schemes discussed above has been implemented in Java, for demonstration purposes (approximately 2,000 LOC). The implementation includes polynomial interpolation as well as calculating the roots of polynomials in $\mathbb{Z}_p[x]$. For the latter, the Cantor-Zassenhaus probabilistic algorithm was used, which calculates the roots in $O(n^3 \log(p))$ operations in $\mathbb{Z}_p$ [10]. It is assumed that the polynomial is square-free, i.e. $D_i \neq D_j$. The implementation and source code will be made available upon request.

6 Extensions for Asymmetry

In the symmetric version of data-based access control, the encryption is symmetric: there is no distinction between keys and data in what one already knows. In this sense, any data may serve as part of a key for recovering other data. Also, it may be imposed that two or more keys be known in order to read certain data. Conversely, this also allows one to recover an additional key if one knows the other keys and the data. This is obviously not a universal solution; in some applications this symmetry is undesirable, for example when privacy-sensitive data is used to gain access. Also in traditional encryption, allowing access to data based on a key does not usually imply allowing access to the key based on the data. Privacy-sensitive data and keys may thus need additional protection, by introducing asymmetry in the policies.

There are at least two ways to introduce asymmetry into data-based access control:

- Give the data-items different weights. This can be done by multiple levels of indirect encryption, in which different combinations of data-items lead to different partial keys. The keys can then be combined to reconstruct the main key for decryption. For example, the policy could be that if you have a, b, c you can get d, e, and vice versa. Now we create three fuzzy extractors: one that creates a key from a, b, c, one that creates a key from d, e and one that creates the main key from any one of the aforementioned keys.
- Use the hashes to keep certain parts of the encryption one-way. If the function h mapping data-items to numbers is a hash function, we can assume that it is infeasible to reconstruct D_i from $h(D_i)$. If we do not include certain data-items in the published encrypted data, this directly implies that these data-items can serve only as credentials, and cannot be recovered by means of the secure sketch. Only the hash can be reconstructed. For example, the policy could be that if you have a and b, you can get c, if you have a and c, you can get b, but if you have b and c, you can't get a. If we use hashes and publish only the encryption of b and c, this policy is satisfied.

Note that the one-way property can also be achieved by multiple-level encryption. In the last example, one can again create three fuzzy extractors, in which the two upper-level ones create a key from a and b, and a and c, respectively. These ideas can be starting points for further research.

7 Strengths and Weaknesses

As far as we are aware, this paper is the first to introduce the idea of data-based access control. In a de-perimeterised world, this approach may help in putting the protection as close to the data as possible, i.e. having the data protect themselves. Compared to sticky policies and attribute-based encryption, this approach has the advantage that it does not need to rely on a separate key authority, because the data themselves allow for recovering the appropriate keys.

The fact that the security depends on data-items is both the main strength and the main weakness of this approach. In contrast to attribute-based encryption, we do not need to rely on a trusted key authority in order to distribute keys based on attributes, because in our case the attributes are data, and serve as keys themselves. This also means, however, that we cannot control the uniformity of the distribution of the key material. This means that the security is only as strong as the entropy in the data used. If $k = 2$ and it is easy to guess two of the data-items, the others are not secure anymore. It is therefore important to develop guidelines on which parameters to choose for the schemes (which items to combine in an encryption, the value of k).

In combination with fuzzy extractors [1], the approach may allow for retrieving data-sets of which items are only approximately known (each item may be slightly wrong). The fuzzy extractor is then used as the function h mapping data-items to numbers. For example, I may supply four passwords to recover my fifth one, and I am allowed make an error of 1 character in each of the passwords. Of course, this comes at the cost of some entropy loss. Also, Reed-Solomon error correction may be used to introduce the property that the *set* of required items is only approximately known (some items may be completely wrong). For example, I may supply four passwords, and if three of them are correct, I get all my five passwords.

After downloading the encrypted data-set, all computations can be done offline, without the need of a third party. Still, it may be possible that an adversary learns the value of a certain $h(D_i)$ without knowing D_i. For protection against this threat, one could use a salt in the function h, to make sure that $h(D_i)$ is different for each data-set the data-item is included in. Whether h could also be a probabilistic function may be a question for further research.

Moreover, the schemes introduced here may not be the most efficient. Research is therefore needed into alternative schemes and comparison of their properties in terms of security and efficiency. This research should also address the scalability of the different methods. In this paper, the aim was only to provide the framework and a proof-of-concept.

8 Conclusions

In this paper, we introduced the concept of data-based access control. This is a generalised form of de-perimeterised security in which the possibility for recovering protected data depends on the data one already knows. More specifically, we introduced the problem of recovering n data-items if one knows k of them, and not learning anything if one knows fewer than k. This problem is similar to a secure sketch, but it does

not inherently involve error correction, and the application area is different. We also introduced possible solutions to this problem based on polynomial interpolation and secure sketches, and suggested some applications. Further research in this new paradigm should reveal alternative and possibly more efficient schemes, as well as additional applications.

In the future, it would be useful to define other forms of data-based access control than (k, n) threshold schemes. Also, it should be investigated how applications, such as password recovery, can be implemented securely in practice. Another open question is how to deal with propagation of access: if an additional data item is recovered, this may in its turn give access to more data items. This may also affect revocation of policies: how can one be sure that users that should not have access anymore do not possess all the required data to gain access? Intuitively, one may add . Further research will investigate how this affects policies.

In this paper, we have focused on confidentiality as a security property. It may also be possible to use data-based security for integrity purposes, in the form of data-based signatures. An additional question that needs to be addressed is how signing with a certain key proves knowledge of the associated data.

Acknowledgements. This research is supported by the research program Sentinels (www.sentinels.nl). Sentinels is being financed by Technology Foundation STW, the Netherlands Organization for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs. The authors wish to thank Pieter Hartel, André van Cleeff and Trajce Dimkov for useful comments on drafts of this paper.

References

1. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
2. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM Journal on Computing 38(1), 97–139 (2008)
3. Ellison, C., Hall, C., Milbert, R., Schneier, B.: Protecting keys with personal entropy. Future Generation Computer Systems 16, 311–318 (2000)
4. Jericho Forum. Jericho whitepaper. Jericho Forum, The Open Group (2005)
5. Juels, A., Sudan, M.: A fuzzy vault scheme. Designs, Codes and Cryptography 38(2), 237–257 (2006)
6. 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)
7. Mannan, M., van Oorschot, P.C.: Digital objects as passwords. In: 3rd USENIX workshop on hot topics in security (2008)
8. Sahai, A., Waters, B.: Fuzzy identity based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
9. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
10. Shoup, V.: A computational introduction to number theory and algebra, 2nd edn. Cambridge University Press, Cambridge (2008)

Improving Cut-and-Choose in Verifiable Encryption and Fair Exchange Protocols Using Trusted Computing Technology

Stephen R. Tate[1] and Roopa Vishwanathan[2]

[1] Dept. of Computer Science, University of North Carolina at Greensboro Greensboro, NC 27402
srtate@uncg.edu
[2] Dept. of Computer Science and Engineering, University of North Texas Denton, TX 76203
rv0029@unt.edu

Abstract. Cut-and-choose is used in interactive zero-knowledge protocols in which a prover answers a series of random challenges that establish with high probability that the prover is honestly following the defined protocol. In this paper, we examine one such protocol and explore the consequences of replacing the statistical trust gained from cut-and-choose with a level of trust that depends on the use of secure, trusted hardware. As a result, previous interactive protocols with multiple rounds can be improved to non-interactive protocols with computational requirements equivalent to a single round of the original protocol. Surprisingly, we accomplish this goal by using hardware that is not designed for our applications, but rather simply provides a generic operation that we call "certified randomness," which produces a one-way image of a random value along with an encrypted version that is signed by the hardware to indicate that these values are properly produced. It is important to stress that while we use this operation to improve cut-and-choose protocols, the trusted operation does not depend in any way on the particular protocol or even data used in the protocol: it operates only with random data that it generates. This functionality can be achieved with minor extensions to the standard Trusted Platform Modules (TPMs) that are being used in many current systems.

We demonstrate our technique through application to cut-and-choose protocols for verifiable group encryption and optimistic fair exchange. In both cases we can remove or drastically reduce the amount of interaction required, as well as decrease the computational requirements significantly.

1 Introduction

Zero-knowledge proofs were first introduced by Goldwasser et al. [19] and have applications in a wide range of cryptographic protocols that require authentication of one party to another. A zero-knowledge proof is a protocol in which a prover P provides some information I to a verifier V, and then engages in a protocol to convince V that I satisfies some property $Q(I)$ that would be

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 252–267, 2009.

difficult for V to compute on its own. For example, I might be an encrypted value and the property $Q(I)$ refers to a simple property of the corresponding plaintext. Zero-knowledge proofs are used in higher-level protocols such as fair exchange [1], identification protocols [16], and group signatures [4]. Interactive zero-knowledge proof systems often employ a paradigm called "cut-and-choose" in which the prover answers a series of random challenges given by the verifier. For each challenge, the verifier has at most a 50% chance of getting cheated. With sufficiently many challenges, the chances of a dishonest prover answering all of them correctly, and the verifier getting cheated on all of them is negligible. This requires several rounds of communication between the prover and verifier, and increases the communication costs, thereby decreasing the efficiency of the protocol. Several protocols for the verifiable encryption problem use such zero-knowledge proofs, and this is the core problem that we examine in this paper.

Verifiable encryption is a protocol that actively involves two parties, a prover and a verifier, and passively involves one or more additional parties. In its simplest version, with a single trusted third party T, the prover P encrypts a secret value s that is supposed to satisfy some specific property (e.g., a signature on some message) with the public key of the trusted third party PK_T, and sends the encrypted value $E_{PK_T}(s)$ to the verifier V. The protocol is such that V is convinced that the received ciphertext will decrypt to a value s which satisfies the necessary property, but other than this fact V is not able to discover any additional information about the value s. In a typical application, a honest prover will later reveal the secret, and the trusted party is only involved if the protocol does not complete and V needs to recover the secret without the assistance of P. Verifiable encryption has been used to construct solutions for fair exchange [1,2], escrow schemes [29], and signature sharing schemes [17]. In this paper we solve a generalized, more powerful version known as verifiable group encryption, in which there are multiple semi-trusted parties ("recovery agents" or "proxies") and authorized subsets of agents. The secret is encoded such that it can be recovered only by an authorized subset of recovery agents working together.

One of the more important applications of verifiable encryption is the fair exchange problem: Two parties A and B have values that they would like to exchange (e.g., A has a credit card number and B has a downloadable song), and after executing a fair exchange protocol either both parties receive the value they are entitled to, or neither does. Fair exchange is the central problem in various online transactions, and with e-commerce transactions growing at an ever increasing rate with operations such as online credit card transactions, online stock trading, and e-checks becoming more common than ever, a large number of businesses depend on these transactions being executed fairly.

The Trusted Computing Group, an industry consortium of over 100 companies, has developed specifications for a hardware chip called the Trusted Platform Module (TPM) [22]. TPMs have become common in laptops, business-oriented desktops, and tablet PCs, including those by IBM, Dell, Lenovo, HP, and Toshiba, as well as in some server-class systems such as the Dell R300. TPMs are designed to be very cheap (under $5 each), and are intended to be

easily embedded on the motherboard of a standard system. While major CPU and chipset manufacturers have designed additional trusted computing components (e.g., the Intel LaGrande project), in this paper we simply require the TPM chip. Primarily, the TPM is used for measurement of system parameters (BIOS, hardware elements, operating system, software applications, among others) to provide a measured and protected execution environment, which assures the integrity and trustworthiness of the platform. To support trustworthy reporting of these measurements to outside parties, TPMs sign measurements with "Attestation Identity Keys" (AIKs): keys that cannot exist in usable form outside the TPM, are only used inside the TPM to sign values produced inside the TPM chip itself, and are certified by a designated certification authority known as a PrivacyCA which vouches for the fact that the certified key is indeed a legitimate AIK which can be used only internally to a TPM. The exact process of establishing that a key is an internal-use only AIK relies on a chain of trust from the manufacturer of the TPM, and is beyond the scope of this paper to describe — interested readers are referred to the TPM documentation for details [22]. In addition to operations with measurements and AIKs, TPMs provide a variety of other capabilities, and in this paper we use the TPM's ability to generate random numbers (or at least cryptographically secure pseudo-random numbers), and the ability to perform encryption. In this paper we show how to use the capabilities of a TPM to improve upon existing verifiable encryption protocols and protocols (specifically fair exchange) that build upon verifiable encryption.

1.1 Our Contribution

We investigate ways in which limited trusted hardware, such as TPMs, can be used to generate certain non-interactive zero knowledge proofs and thus improve protocols that use these cut-and-choose constructions. Interestingly, we show that a significant benefit can be gained from constructions that we design around two proposed very generic TPM commands: `TPM_MakeRandomSecret` and `TPM_EncryptShare`. These are simple extensions to the current TPM specification that involve generating a (pseudo)-random number and signing a set of values with an AIK. Using these two commands, we can bring down the communication costs and decrease the number of rounds required in certain interactive zero-knowledge proof systems. We have identified two applications of our scheme: fair exchange protocols and verifiable group encryption. Using our TPM-based protocols, we construct a verifiable group encryption scheme and solve an open problem in Asokan *et al.*'s paper [1]: making their protocol non-interactive and bringing down the cost of the *verifiable escrow* operation. Other applications may also benefit from our techniques, including applications such as group signatures and identity escrow schemes.

2 Related Work

This paper builds upon work in both cryptography and in hardware-assisted security, and in this section we review related work in each of these areas.

2.1 Cryptographic Protocols

Cut-and-choose zero-knowledge proofs are used in many cryptographic protocols that involve authentication while maintaining privacy, such as the Fiat-Shamir identification scheme [16,4], non-malleable commitments [18], concurrent zero-knowledge protocols [3], and verifying secret shuffles [20], among other protocols. Although there have been methods proposed to reduce the round complexity of zero-knowledge proofs and the design of non-interactive zero-knowledge proofs, most protocols that use such methods require us to accept a weaker form of security [8]. In this paper we focus on applications of interactive zero-knowledge proofs in verifiable group encryption and a specific application of verifiable encryption: fair exchange of signatures.

We now expand on the description of verifiable encryption given in the previous section to make precise what is meant by the secret value satisfying some property. Specifically, there is a given relation $\mathbf{R}$, the prover and verifier share a public value x, and the secret s is such that $(x, s) \in \mathbf{R}$. For example, x could be a public message, and s could be a digital signature made by the prover on that message — while the verifier is convinced that the prover has indeed signed the message and provided an encrypted version of this signature as $E_{PK_T}(s)$, the verifier cannot retrieve the actual signature until either the prover provides it later or the recovery agent T decrypts the signature for the verifier (e.g., at a specific "release time").

The exact relation $\mathbf{R}$ depends on the application of verifiable encryption. In some protocols, the secret is simply the pre-image of the public value x under some one-way function $f(s) = x$, meaning that $\mathbf{R} = \{(x, f^{-1}(x))\}$ — this is the case in the work by Asokan *et al.* [1], where f is additionally a homomorphism. Camenisch and Damgard [7] consider a generalized problem in which the relation $\mathbf{R}$ is any relation possessing a 3-move proof of knowledge which is an Arthur-Merlin game, which they call a *Σ-protocol*, and includes all of the relations considered by Asokan *et al.*'s earlier work.

In addition to introducing Σ-protocols, Camenisch and Damgard also expand the problem from the verifiable encryption problem studied by Asokan *et al.* (with a single trusted party), to the verifiable group encryption problem (with n semi-trusted recovery agents, or *proxies*), that we described in the previous section. Camenisch and Damgard's solution [7] is a cut-and-choose based method in which the prover generates two sets of n shares of an intermediate secret, and then encrypts these $2n$ shares. The verifier asks the prover to open half of those encryptions, and in doing so can verify that the prover honestly constructed that set of encryptions. Thus if the prover attempts to cheat on one (or both) sets of shares, this will be discovered with probably of at least $1/2$. Camenisch and Damgard then repeat this process multiple times to decrease the chance of being cheated to a negligible probability.

Fair exchange (of digital signatures) is an important application of verifiable encryption. There have been a number of protocols developed for fair exchange of digital signatures, and a survey paper by Ray outlines many of them [30]. The work by Asokan, Shoup, and Waidner [1], which we mentioned above for the

results on verifiable encryption, provides one of the most widely-referenced and efficient optimistic fair exchange protocols. Synchronizing messages between the parties so that the fair exchange properties are met is challenging, but the heart of this protocol is their solution to the verifiable escrow problem. Thus, in case one of the parties cheats, the other (honest) party can get the signature or the promised item from the third party. As is common in many other fair exchange protocols, this protocol employs a cut-and-choose zero-knowledge proof between the prover and verifier.

2.2 Hardware-Assisted Security for Cryptographic Protocols

The idea of using trusted hardware tokens to improve the security of cryptographic protocols can be traced back to the work by Chaum and Pedersen [12], Brands [6], and Cramer [13] in which *observers*, or smart-cards, or tokens act as intermediaries in financial transactions between a user and a bank. This idea was recently studied in a theoretical setting by Katz [26] in which user-constructed hardware tokens are used as part of a protocol for realizing multiple commitment functionality in the universal composability (UC) framework. Independently, Chandran, Goyal and Sahai [11], and Damgard *et al.* [14] improve on Katz's results by making the token independent of the parties using it, resettable, and relying on general assumptions like trapdoor permutations rather than cryptographic assumptions. More recently Moran and Segev [28] have improved upon all of the above results by requiring that only one of the two parties involved needs to build a tamper-proof token as opposed to the previous results which require that both prover and verifier have to generate their own hardware tokens. Our work is clearly related to this line of work on hardware tokens, but rather than using tokens created by a participant in the protocol we trust a hardware manufacturer to produce a trustworthy TPM that provides generic, non-application and non-user-specific functionality. A similar approach using TPMs for a different problem was taken by Gunupudi and Tate [24], who show how a TPM with some slight modifications can be used to act in a way indistinguishable from a random oracle, which can then be used in multi-party protocols.

Sarmenta *et al.* [32] introduced the notion of virtual monotonic counters, and designed two schemes to implement this concept using TPMs. Sarmenta *et al.* then show how virtual monotonic counters can be used to produce several interesting applications such as digital wallets, virtual trusted storage, and offline payment protocols. In addition, they also introduced the concept of *Count-limited objects* (or *clobs*), which are cryptographic keys or other TPM-protected objects that are tied to a virtual monotonic counter, and are limited in the number of times that they can be used.

Using the foundation created by Sarmenta *et al.*, Gunupudi and Tate [25] applied *clobs* to create non-interactive protocols for oblivious transfer, designing a particularly efficient technique for a set of concurrent k-of-n oblivious transfers. In particular, a large set of general oblivious transfers can be performed using just a single clob, resulting in a very efficient protocol for a wide range of applications that use oblivious transfer, including secure function evaluation (SFE).

3 Preliminaries and Protocol Primitives

In this section we describe the TPM operations we propose in this paper. For constructing our protocols, we use the ability of the TPM to generate random (secret) numbers, create signatures using one of the TPM's Attestation Identity Keys (AIKs), and do basic encryption. Recall that an AIK is a signing key that is certified by a PrivacyCA as usable only inside a TPM, so we trust that values signed by an AIK were correctly and honestly produced (assuming an uncompromised TPM).

To support the cryptographic operations required by the TPM specification, TPMs must be able to perform modular arithmetic, and we use these capabilities to perform operations over a particular cyclic group. In particular, let p and q be primes such that $q|p-1$, and let g be a generator for the subgroup of $\mathbf{Z}_p^*$ of order q. These values can be global for all TPMs, although it is advisable that these values be modifiable in case particular primes are discovered to have undesirable properties. The operations we require of the TPM will be modulo p and modulo q arithmetic, and based on other keys used by the TPM and other considerations p might be 1024 or 2048 bits, with q being 160 bits.

3.1 Secret Sharing Schemes

Secret sharing is a fundamental part of our work, so we review the concepts and establish notation in this section. Specifically, we look at *threshold secret sharing*, in which a secret is divided among a group of n members who have decided on a threshold value, say, k. Each member gets a share of the secret, and only groups of at least k members can reconstruct the secret. Shamir's scheme [34] was the first such secret sharing scheme. Other secret sharing variations, using both thresholds and more general notions of access control structures, include those by Feldman [15], Blakley [5] and Krawczyk [27]. Formally, a threshold secret sharing scheme consists of two functions: SHARE and RECOVER such that:

SHARE$(x, k, n) \rightarrow (s_1, s_2, \ldots, s_n)$**:** Splits a secret $x \in \mathbf{Z}_q$ into n shares, with each $s_i \in \mathbf{Z}_q$, so that the secret can be reconstructed from any k shares but not from any fewer than k shares.

RECOVER$(v_1, v_2, \ldots, v_k, k, n) \rightarrow y$**:** If $v_1, v_2, \ldots, v_k$ are k different shares produced by the SHARE operation, then the value y that this produces is the original secret value (x in the SHARE operation).

3.2 Our Protocol Primitives

The following are two proposed TPM commands which we will use in this paper. While not necessary for the most straightforward applications, we give the capability to attach a "condition" to the certified randomness produced by these commands — the condition may be arbitrary, but a fixed-length hash is passed to the first command and then included in each encrypted share by the second command (e.g., if SHA-1 is used, like many other TPM commands, this would

be a 160-bit parameter). This condition is vital for more involved protocols, such as the fair exchange protocol that we describe in Sect. 4.3, where a party uses the condition to commit to certain values when starting the verifiable encryption process.

`TPM_MakeRandomSecret`$(hCond, k, n) \rightarrow (secHandle, r)$: This operation generates a random $r \in \mathbf{Z}_q$ that will be shared in a k-of-n secret-sharing scheme, and provides the value and a handle to an internal protected storage version of the secret. The value $hCond$ is the hash of a "condition" tied to this random secret, as described above.

`TPM_EncryptShare`$(secHandle, AIKHandle, i, PK) \rightarrow$
$Sign_{AIK}(E_{PK}(hCond \parallel s_i), PK, k, n, g^r, hCond)$: Gives a signed, encrypted version of the i-th share of random secret r (referred to by $secHandle$). If this is called with $i = 1, \ldots, n$, the shares that are encrypted, $s_1, \ldots, s_n$ should be the shares output by $\textsc{Share}(r, k, n)$ for some secret sharing scheme.

Since TPMs support basic modular arithmetic for encryption operations, these operations are relatively easy to implement efficiently using Shamir's shecret sharing scheme [34]. In some applications, such as the optimistic fair exchange protocol that we describe later, we don't need secret sharing at all. Instead, we simply need to be able to encrypt the secret random value using the public key of one or more trusted parties so that the secret can be recovered later if necessary. This is a degenerate "1-of-n" case of the generic operation above, but we describe it here separately as this might be the most useful form of the certified randomness operations.

`TPM_MakeRandomSecret`$(hCond) \rightarrow (secHandle, r)$: Random $r \in \mathbf{Z}_q$ is selected, stored internally with $secHandle$ to refer to it, and returned to the user. $hCond$ is the hashed condition, as before.

`TPM_EncryptShare`$(secHandle, AIKHandle, PK) \rightarrow$
$Sign_{AIK}(E_{PK}(hCond \parallel r), PK, g^r, hCond)$: The previously generated r is bundled with the condition $hCond$ and encrypted for a trusted party whose public key is denoted by PK, and signed by the Attestation Identity Key (AIK).

4 Our TPM-Based Verifiable Group Encryption Protocol

In this section we show how the TPM operations defined above can be used to implement a form of verifiable group encryption, a problem defined by Camenisch and Damgard [7]. In verifiable group encryption, there is a sender (prover), a receiver (verifier), and n semi-trusted parties ("proxies"). The sender has a public value x that it claims satisfies some property, which can be verified using a secret witness s known to the sender. The sender wants to send x along with additional information to the receiver such that the receiver (a) has assurance that it can recover the secret s if it has the cooperation of an authorized subset of the proxies and (b) without the cooperation of an authorized subset of proxies the receiver

gets no information about s. There are several ways to define an "authorized subset" of proxies, but the most generic way is to use a monotone access structure Γ, which is a set of subsets of authorized proxies that is monotone (so that if $A \in \Gamma$ and $A \subseteq B$ then $B \in \Gamma$).

We next provide a precise and formal definition of this problem. While Camenisch and Damgard formally defined (non-group) Verifiable Encryption and then informally described the group encryption problem [7], they did not formally define the Verifiable Group Encryption problem. Therefore, while our definition is based on their work, the definition given here is new.

While expressed formally, the properties in the definition have simple intuitive descriptions: The *Completeness* property states that if the prover and verifier are honest, the verifier always accepts. The *Validity* property says that no dishonest prover can trick a verifier into accepting something that will not allow recovery of a valid witness s if the verifier uses a set of proxies that are included in the access control structure Γ. The *Computational Zero Knowledge* property states that no dishonest verifier can recover any information about a valid witness s, and this is true even if the verifier uses an arbitrary set of proxies that is not in the access control structure.

Definition 1 (Generic Verifiable Group Encryption). *Let* $\mathbf{R}$ *be a relation and define language* $L_{\mathbf{R}}$ *by* $L_{\mathbf{R}} = \{x \,|\, \exists s : (x, s) \in \mathbf{R}\}$. *A* Verifiable Group Encryption Scheme *for relation* $\mathbf{R}$ *consists of a two-party protocol* (P, V) *and a recovery algorithm* R. *Assume we have a set of* n *proxies with encryption/decryption algorithms denoted* (E_i, D_i) *for* $i = 1, \ldots, n$, *and denote this set with notation* $(\mathcal{E}, \mathcal{D}) = ((E_1, E_2, \ldots, E_n), (D_1, D_2, \ldots, D_n))$. *Let* $V_P(\mathcal{E}, x, \Gamma, \lambda)$ *denote the output of the verifier when interacting with* P *on input* $(\mathcal{E}, x, \Gamma, \lambda)$, *where* Γ *is a monotone access structure over* $\mathcal{D}$ *and* λ *is a security parameter. Let* $\perp$ *denote the null set. The following properties must hold:*

1. *Completeness: If* P *and* V *are honest, then for all sets of proxies* $(\mathcal{E}, \mathcal{D})$ *and all* $x \in L_{\mathbf{R}}$,
$$V_P(\mathcal{E}, x, \Gamma, \lambda) \neq \perp \ .$$
2. *Validity/Soundness: For all polynomial time* $\tilde{P}$, *all* $(\mathcal{E}, \mathcal{D})$, *all* Γ, *all* $\tilde{\mathcal{D}} \in \Gamma$, *all positive polynomials* $p(\cdot)$, *and all sufficiently large* λ, *if* $\alpha = V_{\tilde{P}}(\mathcal{E}, x, \Gamma, \lambda)$ *and* $\alpha \neq \perp$ *then*
$$Prob[(x, R(\tilde{\mathcal{D}}, \alpha)) \notin \mathbf{R}] < 1/p(\lambda) \ .$$
3. *Computational Zero Knowledge: For any polynomial time* $\tilde{V}$, *which is a verifier that takes a subset of decryption proxies* $\tilde{\mathcal{D}} \subseteq \mathcal{D}$ *in addition to the regular verifier parameters, there exists an expected polynomial time simulator* $S_{\tilde{V}}$ *with black-box access to* $\tilde{V}$ *such that for all polynomial time distinguishers* A, *all positive polynomials* p, *all* $x \in L_{\mathbf{R}}$, *and all sufficiently large* λ, *if* $\tilde{D} \notin \Gamma$ *we have*
$$Prob[A(\mathcal{E}, x, \alpha_i) = i : \alpha_0 = S_{\tilde{V}}(\mathcal{E}, x, \lambda);\ \alpha_1 = \tilde{V}_P(\mathcal{E}, \tilde{D}, x, \lambda);\ i \in_R \{0, 1\}]$$
$$< \frac{1}{2} + \frac{1}{p(\lambda)} \ .$$

Definition 1 is a generic form of verifiable group encryption, but in order to produce algorithms that can be put into fixed trusted hardware, we restrict two general parameters in the above definition — we fix the relation $\mathbf{R}$ to be a specific relation based on the discrete log problem, and we restrict the access control structure Γ to be a threshold structure so that we can use simple threshold secret sharing schemes. Specifically, we make the following two restrictions:

1. $\mathbf{R} \subseteq (\mathbf{Z}_p, \mathbf{Z}_q)$ is defined so that $\mathbf{R} = \{(g^s, s) \,|\, s \in \mathbf{Z}_q\}$. In other words, in any pair $(x, s) \in \mathbf{R}$, the secret s is the discrete log of x.
2. $\Gamma = \{\tilde{\mathcal{D}} \,|\, \tilde{\mathcal{D}} \subseteq \mathcal{D} \text{ and } |\tilde{\mathcal{D}}| \geq k\}$

Since we will be using trusted platforms to implement verifiable group encryption, we need to establish the security properties of a trusted platform so that we can reason about the security of our implementation. This assumption is nothing more than an explicit description of what it means for a TPM to be secure in the obvious sense, and reflects the requirements given in the TPM Protection Profile [21]. While no hardware can be perfectly protected, currently manufactured TPMs appear to have a high level of tamper resistance with regard to the protection of secrets, and we believe that our security assumption is realistic.

Definition 2. *The* Trusted Platform Security Assumption *is the assumption that the system containing a TPM satisfies the following properties:*

1. Tamper-resistant hardware: *It is infeasible to extract secrets stored in protected locations in the TPM.*
2. Secure Encryption: *The public-key encryption algorithm used by the TPM is CCA-secure.*
3. Secure Signatures: *The digital signature algorithm used by the TPM is existentially unforgeable under adaptive chosen message attacks.*
4. Trustworthy PrivacyCA: *Only valid TPM-bound keys are certified by a trusted PrivacyCA.*

Assuming we have a trusted platform that provides the operations defined in Sect. 3.2 and satisfies the Trusted Platform Security Assumption, we define the following protocol for the verifiable group encryption problem — since our protocol is non-interactive, we define it as a pair of algorithms, one for the sender which produces the verifiably-encrypted secret, and one for the receiver which verifies that the values produced by the sender are properly formed.

In the following algorithms, $PK_1, \ldots, PK_n$ denote the public encryption keys of the n proxies, so the n-tuple $(PK_1, \ldots, PK_n)$ is the realization of the abstract set of encryption routines $\mathcal{E}$ given in Definition 1. Furthermore, since the abstract access structure Γ is restricted to be a threshold structure, it is fully specified by the pair (k, n), which represents a "k-of-n" threshold structure. Finally, we assume that the AIK used in this protocol is loaded into the TPM before VESender is called and is referenced by TPM handle $AIKH$, and we have a certificate $Cert(AIK)$ for this AIK signed by a trusted PrivacyCA. The condition $hCond$ associated with this escrow is the same as described in Sect. 3.2, and can be used as needed by applications.

Algorithm. VESENDER$((PK_1, \ldots, PK_n), s, (k, n), hCond, \lambda)$
$x \leftarrow g^s$
$(secHandle, r) \leftarrow$ `TPM_MakeRandomSecret`$(hCond, k, n)$
$d \leftarrow s + r \bmod q$
$t \leftarrow g^r \bmod q$
for $i \in 1, \ldots, n$ **do** $C_i \leftarrow$ `TPM_EncryptShare`$(secHandle, AIKH, i, PK_i)$
return $\mathcal{B} = \langle d, x, t, C_1, C_2, \ldots, C_n, Cert(AIK)\rangle$

Algorithm. VERECVERIFY$(\mathcal{B} = \langle d, x, t, C_1, C_2, \ldots, C_n, Cert(AIK)\rangle, hCond)$
if $Cert(AIK)$ is not certified by a trusted PrivacyCA **then return** $\perp$
if $C_1, \ldots, C_n$ are not tuples $\langle c_i, PK_i, k, n, t, h\rangle$ signed by AIK **then return** $\perp$
if $\langle k, n, t, h\rangle$ are not identical in all tuples with $h = hCond$ **then return** $\perp$
if $g^d \neq xt$ **then return** $\perp$
return $\alpha = \mathcal{B}$

If it is necessary to use these encryptions to recover the secret s, the following algorithm can be used:

Algorithm. VERECOVER$(\alpha = \langle d, x, t, C_1, C_2, \ldots, C_n, Cert(AIK)\rangle, hCond)$
Select k C_i's: $C_{i_1}, C_{i_2}, \ldots, C_{i_k}$, and use proxies to decrypt each $hCond_{i_k} \parallel s_{i_k}$
if any $hCond_{i_k}$ does not match parameter $hCond$ **then return** $\perp$
$r \leftarrow$RECOVER$(s_{i_1}, s_{i_2}, \ldots, s_{i_k}, k, n)$
$s' \leftarrow (d - r) \bmod q$
return s'

Since the TPM guarantees that the recovered r is such that $t = g^r$, and we have verified that $g^d = xt$, we have $g^d = xg^r$, so s' is such that $g^{s'} = g^{d-r} = x$. Since s is the unique value in $0, \ldots, q-1$ such that $g^s = x$, we must have $s' = s$, and we have recovered the original secret s. Below we demonstrate a useful application of this scheme.

4.1 Application to Verifiable Group Encryption of Signatures

In this section, we show how to apply our verifiable group encryption protocol to the problem of verifiably encrypting shares of a valid digital signature. In this problem there is a publicly-known message m, and we would like for a party with public key y to sign this message. The signature is not to be revealed immediately, but needs to be committed to and escrowed in such a way that an appropriate set of trusted parties can recover the signature if necessary. We would like to verifiably encrypt the signature so that the receiver has assurance that the ciphertexts which it receives (and which are unintelligible to it) are in fact shares of the requested signature.

Our verifiable encryption protocol can be used with many signature schemes, including GQ [23], Fiat-Shamir [16], and RSA [31], but for concreteness we instantiate it here with Schnorr signatures [33]. Techniques for using other signature schemes are similar to the techniques described in Section 4 of Asokan *et al.* [1]. Note that the verifiable encryption protocol is independent of the signature scheme used, and the security of the verifiable encryption is in no way related to the security of the underlying signature scheme. The Schnorr signature scheme is briefly described below:

SETUP OF SYSTEM PARAMETERS: There is a prime modulus p with a generator g of a subgroup of $\mathbf{Z}_p^*$ of prime order q, where q is also prime. A random value $\zeta \in \mathbf{Z}_q$ is the private key of the signer, Alice, and her public key is $y = g^\zeta$. Let Bob be the verifier.

SCHNORRSIGN(m, ζ): To create her signature, Alice picks a random $r \in \mathbf{Z}_q$ and computes $u = g^r$, $c = h(u \parallel m)$, and $z = r + \zeta c \bmod q$, where h is a cryptographic hash function. Her signature is then (c, z).

SCHNORRVERIFY$((c, z), m, y)$: Bob, who knows Alice's public key y, checks if $h(g^z y^{-c} \parallel m) = c$, and accepts if it is true, rejects otherwise.

Next we show the sender and receiver portions of our verifiable group encryption of a Schnorr signature. Since the Schnorr signature is a pair (c, z), where c is random (in the random oracle model), we transmit c in the clear and use the verifiable encryption for only the second component z.

Algorithm. SIGSENDER$((PK_1, \ldots, PK_n), m, (k, n), \zeta, hCond)$
$(c, z) \leftarrow$SCHNORRSIGN(m, ζ)
$\mathcal{B} \leftarrow$VESENDER$((PK_1, \ldots, PK_n), z, (k, n), hCond, \lambda)$
return $\langle (c, z), \mathcal{B} \rangle$

Algorithm. SIGVERIFIER$(y, m, c, \mathcal{B} = \langle d, x, t, C_1, C_2, \ldots, C_n, Cert(AIK) \rangle, hCond)$
if y is not an acceptable public key **then return** $\perp$
if $h(xy^{-c} \parallel m) \neq c$ **then return** $\perp$
return VERECVERIFY$(\mathcal{B}, hCond)$

If it is necessary to recover the signature from the encrypted shares, we first do a recovery from the verifiable group encryption to recover z. Since SIGVERIFIER has verified that $h(xy^{-c} \parallel m) = c$, and the recovered z is such that $x = g^z$, we have $h(g^z y^{-c} \parallel m) = c$, which is exactly the property that must be verified in SCHNORRVERIFY. Therefore, (c, z) is a valid Schnorr signature on message m.

Algorithm. SIGRECOVER$(c, \mathcal{B}, hCond)$
$z \leftarrow$ VERECOVER$(\mathcal{B}, hCond)$
if $z = \perp$ **then return** $\perp$
return (c, z)

Our building blocks for this protocol, VESENDER and VERECVERIFY, provide a secure verifiable group encryption scheme, reflected in the following theorem.

Theorem 1. *Let* $\mathbf{R}$ *be a relation such that* $\mathbf{R} \subseteq (\mathbf{Z}_p, \mathbf{Z}_q)$ *is defined so that* $\mathbf{R} = \{(g^z, z) \mid z \in \mathbf{Z}_q\}$. *The protocol outlined above, when the prover uses a system that satisfies the Trusted Platform Security Assumption, is a secure verifiable group encryption scheme for* $\mathbf{R}$.

Proof: Due to space limitations we have removed this proof from the conference paper. It can be found in the full version [35].

4.2 Comparison to Non-trusted Platform Protocols

We now briefly compare the costs of our protocol to two verifiable encryption protocols: that of Camenisch and Damgard [7] (the "CD" protocol), and Camenisch and Shoup [9] (the "CS" protocol), in Table 1. Note that the CS protocol is described only for verifiable encryption of a single secret (not verifiable group encryption); in the table, we estimate the values for k shares.

Although we cannot fully describe these protocols here due to lack of space, we note that our full protocol is roughly comparable to one round of the CD protocol and is completely non-interactive as compared to the CS protocol. The CD protocol must be repeated in order to build trust: a dishonest prover can get away with cheating on a single iteration of the secret-sharing phase with probability $1/2$, and this probability is reduced to $1/2^R$ by repeating the protocol R times. Using 30 rounds means the probability of the prover cheating without detection is about one in a billion, and a very high degree of assurance can be obtained by repeating the protocol 80-100 times.

The CS protocol improves on the CD protocol, but is not completely non-interactive, although it reduces the amount of interaction from that required by the CD protocol. In addition to providing improved efficiency, our protocol is completely non-interactive: the prover computes some values, sends them off, and then is no longer involved. In an e-mail setting (send and forget), this is a vital property that our protocol achieves but earlier solutions do not.

4.3 Fair Exchange of Digital Signatures

Fair exchange is an important application of the verifiable group encryption protocol outlined above, where additional operations are included to synchronize the actions to ensure that the fair exchange property is satisfied. One of the open problems left by Asokan *et al.* [1] was to come up with a way to eliminate the expensive verifiable escrow operation and make their protocol non-interactive. By using TPMs, we have shown that we can indeed make the protocol non-interactive, greatly improving efficiency.

Table 1. Comparison of the cost of the CD [7] and CS [9] protocols to our TPM-based solution for n proxies

CD protocol		CS protocol	Our Solution
Interactive		Interactive	Non-interactive
Cost for *One Round* (of 30-100)		Cost for Full Solution	Cost for Full Solution
Prover:	2 n-party SHAREs	1 n-party SHARE	1 n-party SHARE
	$2n$ encryptions	n encr.	n encr.
		n signatures	n sig.
Verifier:	n encrytions	n sig. verifies	$n+1$ sig. verifies
	1 or 2 mod powerings	1 to 4 mod pow.	1 mod pow.
	0 or 3 mod multiplies	1 to 4 mod mult.	1 mod mult.
Recovery:	k decryptions	k decr.	k decr.
	1 secret sharing RECOVER	1 sec. shar. RECOVER	1 sec. shar. RECOVER

In fair exchange of signatures, two parties have messages that they have pledged to sign, and at the end of the protocol either both have received the other's signature, or neither has. For concreteness, we assume Schnorr signatures (as we used in our SigSender function), but this can easily be adapted to any signature scheme that has a secure reduction scheme using the homomorphism $\theta(x) = g^x$. At the beginning of the protocol, parties A and B have agreed to sign messages m_A and m_B, respectively, and along with the publicly-known messages both parties know each other's signature public keys y_A and y_B (corresponding private keys ζ_A and ζ_B are known only to A and B, respectively). Asokan *et al.*'s protocol is an "optimistic fair exchange protocol," meaning that the trusted third party is not involved unless there is a dispute, but the third party does require a single, consistent database of tuples to keep track of any requests that have been made of it. The generality of our solutions for verifiable encryption in this paper, using multiple trusted parties and a threshold scheme, causes complications for the fair exchange problem — unless a single globally-consistent database is maintained, it might be possible for a party to cheat the controls that the trusted party is supposed to enforce. While we could potentially do this with some distributed database of tuples, we simplify the problem back to a single trusted party in this section. The trusted party has public encryption key PK, which is known to all parties.

Below we give our modification of Asokan *et al.*'s fair exchange protocol, which has our construct incorporated. In addition to inserting our signature escrow functions SigSender, SigVerifier, and SigRecover, a few changes come from the fact that we are using a specific signature scheme rather than a generic homomorphic reduction scheme.

1. A chooses $r \in \mathsf{Domain}(f)$ at random and computes $v = f(r)$. A then computes its signature σ_A, encrypts it using a regular escrow scheme with condition (v, m_B, y_B), giving an escrow α which it sends along with v to B.
2. B receives v and α from A, and computes $hCond = h(\langle v, \alpha, m_A, y_A \rangle)$, then $\langle (c_B, z_B), \beta \rangle = \text{SigSender}(PK, m_B, \zeta_B, hCond)$. B's signature on m_B is then $\sigma_B = (c_B, z_B)$. B sends c_B and β to A.
3. A receives c_B and β from B, computes it's own copy of $hCond$ and checks that β is a proper signature escrow using $\text{SigVerifier}(y_B, m_B, c_B, \beta, hCond)$ — if this does not verify, A calls $\text{A-Abort}(r, m_B, y_B)$ and quits the protocol (perhaps having received B's signature σ_B); otherwise, A creates signature σ_A for message m_A and sends σ_A to B.
4. B receives σ_A from A and verifies that this is a valid signature on m_A. If this is a valid signature, then B sends signature σ_B to A; otherwise, B calls $\text{B-Resolve}(v, \alpha, m_A, y_A, m_B, y_B, \sigma_B)$ which either returns σ_A (if α is a valid escrow) or an error message.
5. A receives σ_B from B and verifies that this is a valid signature on m_B. If this is a valid signature, then the protocol halts, with both parties having received valid signatures; otherwise, A calls $\text{A-Resolve}(r, \alpha, \beta, c_B, m_B, y_B, m_A, y_A)$ to get σ_B.

Functions A-Abort, A-Resolve, and B-Resolve are executed (atomically) by the trusted party, and are straightforward adaptations of the functions designed by Asokan *et al.* [1] following changes we made in the base exchange protocol above. The details of these functions are provided in the full paper [35].

The fairness of this protocol is established following reasoning similar to that in Asokan *et al.* [1], using Theorem 1 in this paper for the security of our verifiable encryption scheme, resulting in the following theorem. The proof of this theorem is a simple modification to the proof in Asokan *et al.* [1], so is not given here.

Theorem 2. *Given parties A and B, in which B has access to a TPM that satisfies the Trusted Platform Security Assumption, the protocol described above is a secure optimistic fair exchange protocol.*

The most expensive operation in Asokan *et al.*'s protocol is the verifiable escrow operation which makes the cost of the protocol grow with R, where R is the number of rounds between prover and verifier. Using our functions in place of the verifiable escrow, we only need a single round, reducing the computational cost, and perhaps more importantly makes that part of the fair exchange protocol non-interactive.

5 Conclusion and Future Work

We have presented a generic subroutine-like TPM-based construct that we used to create algorithms that replace cut-and-choose protocols in some interactive zero-knowledge proofs, making that part of the protocol non-interactive and more computationally efficient. In the process we have provided an efficient protocol for verifiable group encryption and improved the efficiency of the protocol for fair exchange of signatures due to Asokan *et al.* [1]. Our protocols are generic and independent of the signature scheme being used.

An intestesting open problem is whether other cut-and-choose protocols can also be replaced with the help of the TPM-based techniques developed in this paper, and we are exploring the possibility that our construct could be used to improve other applications. Another interesting direction to pursue would be looking at whether the security our TPM-based techniques can be reasoned about in Canetti's strong universally composable model of security [10]. We believe that this is indeed possible, and could provide results quite similar to Katz's results but using our standard hardware components rather than custom-designed hardware tokens.

References

1. Asokan, N., Shoup, V., Waidner, M.: Optimistic Fair Exchange of Digital Signatures. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 591–606. Springer, Heidelberg (1998)
2. Bao, F., Deng, R., Mao, W.: Efficient and Practical Fair Exchange Protocols with an Off-line TTP. In: Proceedings of the 19th IEEE Symposium on Security and Privacy, pp. 77–85 (1998)

3. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent Non-Malleable Zero Knowledge. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 345–354 (2006)
4. Bellare, M., Palacio, A.: GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and concurrent attacks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 162–177. Springer, Heidelberg (2002)
5. Blakley, G.: Safeguarding Cryptographic Keys. In: AFIPS National Computer Conference, pp. 313–317 (1979)
6. Brands, S.: Untraceable Off-line Cash in Wallets with Observers. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 302–318. Springer, Heidelberg (1994)
7. Camenisch, J., Damgard, I.: Verifiable Encryption, Group Encryption, and their Applications to Separable Group Signatures and Signature Sharing Schemes. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 331–345. Springer, Heidelberg (2000)
8. Camenisch, J., Lysyanskaya, A.: An identity escrow scheme with appointed verifiers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 388–407. Springer, Heidelberg (2001)
9. Camenisch, J., Shoup, V.: Practical Verifiable Encryption and Decryption of Discrete Logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
10. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: IEEE Symposium on Foundations of Computer Science, pp. 136–145 (2001)
11. Chandran, N., Goyal, V., Sahai, A.: New Constructions for UC Secure Computation using Tamper Proof Hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 545–562. Springer, Heidelberg (2008)
12. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers (extended abstract). In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
13. Cramer, R., Pedersen, T.J.: Improved Privcy in Wallets with Observers (extended abstract). In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 329–343. Springer, Heidelberg (1994)
14. Damgard, I., Nielsen, J.B., Wichs, D.: Isolated Proofs of Knowledge and Isolated Zero Knowledge. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 509–526. Springer, Heidelberg (2008)
15. Feldman, P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: FOCS 1987, pp. 427–437. IEEE Computer Society, Los Alamitos (1987)
16. Fiat, A., Shamir, A.: How to prove to yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
17. Franklin, M., Reiter, M.: Verifiable Signature Sharing. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 50–63. Springer, Heidelberg (1995)
18. Gennaro, R.: Multi-trapdoor Commitments and their Applications to Proofs of Knowledge Secure under Concurrent Man-in-the-Middle Attacks (2004)
19. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM Jounal of Computing, 186–208 (1989)
20. Groth, J.: A verifiable Secret Shuffle of Homomorphic Encryptions. In: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography, pp. 145–160 (2003)

21. Trusted Computing Group. Protection profile — PC client specific trusted platform module, https://www.trustedcomputinggroup.org/specs/TPM/ (TPM Family 1.2; Level 2)
22. Trusted Computing Group. Trusted Platform Module Specifications – Parts 1–3, https://www.trustedcomputinggroup.org/specs/TPM/
23. Guillou, L., Quisquater, J.: A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 216–231. Springer, Heidelberg (1990)
24. Gunupudi, V., Tate, S.R.: Random Oracle Instantiation of Distributed Protocols using Trusted Platform Modules. In: 21st International Conference on Advanced Information Networking Applications, pp. 463–469 (2007)
25. Gunupudi, V., Tate, S.R.: Generalized Non Interactive Oblivious Transfer using Count-Limited Objects with Applications to Secure Mobile Agents. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 98–112. Springer, Heidelberg (2008)
26. Katz, J.: Universally Composable Multi Party Computation using Tamper-proof Hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
27. Krawczyk, H.: Secret sharing made short. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 136–146. Springer, Heidelberg (1994)
28. Moran, T., Segev, G.: David and Goliath Commitments: UC Computation for Asymmetric Parties using Tamper-proof Hardware. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 527–544. Springer, Heidelberg (2008)
29. Poupard, G., Stern, J.: Fair Encryption of RSA Keys. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 172–190. Springer, Heidelberg (2000)
30. Ray, I., Ray, I.: Fair Exchange in E-Commerce. In: Proceedings of ACM SIGEcom Exchange, pp. 9–17 (2002)
31. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 120–126 (1978)
32. Sarmenta, L.F.G., van Dijk, M., O'Donnell, C.W., Rhodes, J., Devadas, S.: Virtual Monotonic Counters and Count-Limited Objects using a TPM without a Trusted OS. In: STC 2006 (2006)
33. Schnorr, C.: Efficient Signature Generation by Smart Cards. Journal of Cryptology, 161–174 (1991)
34. Shamir, A.: How to share a secret. Communications of the ACM, 612–623 (1979)
35. Tate, S.R., Vishwanathan, R.: Improving cut-and-choose in verifiable encryption and fair exchange protocols using trusted computing technology, http://eprint.iacr.org/

PAES: Policy-Based Authority Evaluation Scheme

Enrico Scalavino, Vaibhav Gowadia, and Emil C. Lupu

Department of Computing, Imperial College London
{escala,vgowadia,e.c.lupu}@imperial.ac.uk

Abstract. *Enterprise Rights Management (ERM)* systems aim to protect disseminated data even after it has been sent to remote locations. Existing systems are based on common components, have similar functionalities and often have two shortcomings: a centralised architecture and a lack of concern for the trust and privacy of data recipients. To access the data, recipients must present their credentials to a policy evaluation authority, which they cannot choose and may not trust. Furthermore, recipients may be unable to access the data if their connection is intermittent or if they are off-line. To address these limitations, we propose *PAES: a Policy-based Authority Evaluation Scheme*, which combines data protection with a distributed policy evaluation protocol. The result allows us to implement the sticky policies paradigm in combination with trust management techniques. This permits distributing policy evaluation over a flexible set of authorities, simultaneously increasing the resilience of policy enforcement.

1 Introduction

Organisations and individuals gather and exchange data on a daily basis to conduct their business and their life. Information about customers, users, collaborators and competitors is gathered and stored; new data is produced through business activities or received from third parties. This data is often exchanged with clients and collaborators and needs to be adequately protected even after it has been delivered to them.

This problem has been gaining increasing attention in recent years. Controlling data usage after dissemination to remote parties is a challenge which underpins Enterprise Data Sharing, Privacy and Digital Rights Management (DRM). Both academia and industry have proposed several approaches to the problem, which is often referred to in industry as *Enterprise Rights Management* (ERM). Although these approaches vary on aspects such as policy/rights deployment and data protection mechanisms, common design principles and functionalities can be identified. First, data is associated with access rights (or usage control policies) and cryptographically protected before distribution to recipients. Then, a central *trusted authority (TA)*, often the originator himself, responds to access requests, evaluates recipients' rights and issues the decryption keys. A trusted component running on the recipient devices that we will refer to as the *virtual machine (VM)*, ensures that rights are locally enforced. Specific architectures differ in their management of user authentication, policy and rights retrieval, audit of user actions and other tasks [4,16,17,18,23].

Although used in many ERM products and in research, this architecture has several limitations. First, the set of TAs is statically and explicitly defined. Changes in this set,

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 268–282, 2009.

such as adding a new TA or removing an existing one, require defining new agreements, establishing new trust relationships and implementing them in terms of key exchanges, trust records etc. A second issue concerns the availability of the TA: if recipients cannot contact the specified TA, they cannot access the data received. Furthermore, current protocols assume that recipients are willing to present their credentials to the authority designated by the data originator even when credentials may be confidential. A last issue arises in particular in organisational environments where partners do not know the internal structure of the organisation with whom they interact. In such situations, it is difficult for the originator to specify access rules based on internal roles or to specify which entities in the partner organisation are entitled to issue or verify specific credentials. Instead, recipients are typically better placed to choose which authority could better evaluate their credentials amongst a scope of *acceptable* authorities defined by the originator.

To address these issues we propose a *Policy-based Authority Evaluation Scheme (PAES)*. Its design is motivated by the idea that authority to evaluate policies can be granted by other policies associated with them in the same manner as access to data is granted by policies associated with the data. Therefore, in addition to specifying the criteria recipients must meet to access the data, PAES also permits data originators to specify the criteria that authorities must meet to be trusted to evaluate policies. The set of TAs is therefore a dynamic set defined by characterisation and this confers increased flexibility in finding authorities mutually trusted by both data originators and recipients.

In PAES any entity that satisfies the originator's requirements can act as a policy evaluation authority. The same principle can be generalised to attribute and identity certification. In PKI infrastructures, authorities certify the association of an identity (or attribute) to a public key. Users trust certificates issued by authorities but authorities issue certificates according to their own policies and practices. Thus certificates from different authorities have different meanings and varying confidence in the associations they certify. In PAES, authorities are only trusted to evaluate the policies they receive and are designated by other policies. Therefore, authority hierarchies can be built according to the user's policies and the certificates issued carry both the authority signature and references to the policies they are based on. Recipients of a certificate can thus verify whether an authority is part of a specific hierarchy but also whether the policy used by the authority satisfies their requirements. This is possible in so-called policy-based PKIs [13], but the policies included in certificates are still decided by the authorities, not the users.

In this paper, we aim to describe design principles that can lead to more flexible data dissemination control and address the issues discussed above. Therefore, we first present PAES as a general approach before proposing an implementation. The rest of the article is organised as follows: related work is presented in Section 2 whilst Section 3 describes an application scenario providing a context for the examples; Section 4 gives a general overview of our approach while Section 5 describes the protocol and the data protection mechanism we propose to implement PAES. Section 6 introduces a possible policy language to represent PAES policy chains. A general discussion on the proposed solutions is given in Section 7. Finally, conclusions are drawn in Section 8, which also briefly discusses future work.

2 Related Work

Park et al. [18] proposed guidelines for ERM architecture design whose central concepts are the *virtual machine (VM)*, *control set* and *control center*. Virtual machines are software components running on the recipient's devices. Control sets are lists of usage policies or rights the VM enforces. Control centers are similar to what we previously called TAs. Most existing solutions comprise these components and vary on the configuration, responsibilities of these components and their implementation in software or in hardware.

Most ERM products, such as the Authentica [4], Liquid Machines [2] or Microsoft RMS (Rights Management System) [16] originate in industry. MS RMS is perhaps representative of their design. Its architecture is centralised and based on the deployment of publishing servers that broadly correspond to TAs and that issue encryption keys to users authorised to disseminate data and decryption keys to users authorised to access it. Before disseminating data, users *publish* it on a publishing server. The server then creates a *publishing licence* that contains an Access Control List for the data, a reference to the publishing server and the key used to encrypt the data (protected for the server itself). The user can freely distribute the licence to recipients that use it to contact the publishing server and obtain the access rights and decryption keys protected for each recipient in a *use licence*. Other systems differ mainly in the expressiveness of the authorisation policies, the authentication mechanisms employed, the policy deployment methods and the techniques used to store credentials.

Different research proposals have also been made. Yu et al. [24] presented the *Display-Only File Server (DOFS)* where data processing is performed on remote trusted servers and clients only receive snapshots, i.e. partial views of the actual data. The principle behind this design is that the actual data never leaves the trusted server. However, the solution presents similar shortcomings to existing systems as snapshots are only a subset of the original data with different formatting.

More recently, several studies have focused on trusted computing (TC) [22,10] and its applicability to ERM systems. Sandhu et al. [20] defined a family of Policy, Enforcement and Implementation (PEI) frameworks. TC is used to seal the decryption keys for the data into trusted VMs. VMs are then trusted to correctly authenticate recipients and to not disclose the keys. The authors later extended the proposed models allowing user-based access control policies and describing a specific implementation [21].

Casassa Mont introduced the use of sticky policies and of Identity Based Encryption (IBE) [7] [19] to transform policies into encryption keys and bind them to the encrypted data [17]. Decryption keys can then be issued only by a specific TA, chosen at encryption time, if the policies are satisfied. This approach avoids the data publication phase as all users know the TA's public parameters and can encrypt and distribute data. The Attribute-Based Encryption (ABE) scheme [8] [11] can be considered an evolution of IBE use, where decryption keys are generated through any credentials combination that satisfies the policy. With respect to IBE, ABE avoids contacting a TA for policy evaluation and key issuing, i.e. it enables off-line data access.

In contrast to the work described above, our approach aims to apply in ERM concepts derived from trust management frameworks such as the SPKI/SDSI [9], the RT [15] family of languages and SecPAL [6] [5]. More specifically we use the principle

that authority over a decision (e.g. an attribute assignment) can be delegated to entities satisfying specific criteria. However, when used in the context of ERM, the systems mentioned above have a number of shortcomings. First, their evaluation model relies on a central evaluator to which all the credentials must be presented. This extends the problem of credential confidentiality to all the entities in the chain. Second, the policies used for attribute assignment are not decided by the originator who thus loses control over the delegation sequence. In contrast, in PAES: i) entities are only delegated the right to evaluate pre-defined policies, ii) authority over a policy evaluation cannot be further delegated, iii) authority is granted by the positive evaluation of another policy itself evaluated by an authorised entity. Intuitively, this recursive process generates an evaluation/authority chain similar to those introduced by Trust Management systems.

3 Application Scenario

We consider two fictional organisations, the Sacred Heart Hospital (SHH) and the Medical Research Centre (MRC), co-operating to develop a new drug. MRC develops new drug formulations while SHH runs trials with patients and feeds back the results. The process is iterative so trial results affect new drug formulations and information is exchanged between the two partners at each iteration. MRC needs to access the records of patients involved in the trial to assess their response but also to review symptoms, side-effects and potential interference with other medications. SHH needs access to drug formulations to evaluate potential effects on the patients' health. Both SHH and MRC manage confidential data. SHH manages medical records governed by legislation such as HIPAA in US. MRC could abuse access to these records to influence the trials' outcomes or to advertise other drugs. On the other hand, MRC's data on drug formulations and their effects on patients is commercially sensitive. In this context, successes represent a competitive advantage while failures and negative side-effects are a source of embarrassment. Therefore both confidentiality and integrity of the data must be protected regardless of the administrative domain in which the data resides.

Figure 1 shows a particular interaction in this scenario where patients' case histories are distributed to personnel working on the project. In particular we consider SHH (and its doctors) as the data originators and MRC biochemists as the data recipients.

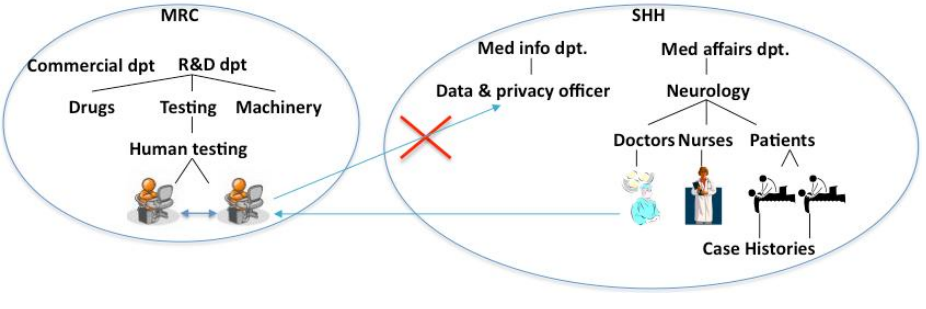


Fig. 1. Cooperative Scenario Example

Existing ERM frameworks (see Section 2) would typically force biochemists to present their credentials to a precise TA in SHH (e.g., the data protection officer) to obtain access rights to the data. This is not suitable for two reasons. First, the SHH data protection officer must authenticate data recipients ensuring their role and participation in the project. For this, it should have knowledge of MRC staff or of MRC authorities on this matter. Second, biochemists must agree to send their credentials (e.g. including on all their skills and competencies) to the data protection officer in SHH that they may not trust with their personal information. Even if the effect of credential disclosure could be mitigated through the use of trust management systems, this solution in not viable in our scenario because of the many shortcomings already described in Section 2. For example, a trust management policy may specify that whoever is trusted as a biochemist by MRC, should have access to the patient case file. The patient however has no control or knowledge of the criteria MRC uses in its trust decisions. MRC might delegate them to a third entity (e.g. an employment agency for scientists, SEA) the patient neither knows nor trusts. When a biochemist, Alice, receives the data her credentials as a biochemist are certified by SEA, which is trusted by MRC, but they must still be presented to an authority in SHH for evaluation. This violates not only Alice's privacy, but also reveals the business relationship between SEA and MRC. Some ERM systems such as MS RMS may allow SHH to define a TA internal to MRC. However, this does not mean that biochemists necessarily trust this TA with their personal information. In fact, they may even not trust TAs defined by MRC itself. Moreover, cross-organisational trust relationships must be set up statically. In case of intermittent or absent connectivity, or unavailability of the designated TA, biochemists cannot access the data.

PAES aims to allow MRC biochemists to be authenticated through any chain of authorities whose root is directly trusted by the data originator and to choose any entity they trust to evaluate them, provided that it meets some specified criteria. We present a more detailed example in Section 4.

4 The Policy Authority Evaluation Scheme

An *access (or usage) control policy* defines a set of criteria that must be satisfied to perform a specific action. Policies expressed and enforced by existing languages and control systems [1,3,12,14] mostly convey this meaning. Therefore, each policy comprises a target object, an action and a set of access conditions and can be defined as:

Definition 1. *A policy p is a tuple (t,a,c) where t is a protected target object, a is an operation that can be executed on the object and c denotes a set of conditions constraining the operation execution.*

Let R_p be the set of entities satisfying policy p, i.e. that can perform action a on object t. Note that the set changes continuously as the conditions in c can be either satisfied or not at different moments in time.

Definition 2. *An Evaluation Authority EA for policy p is an entity trusted to evaluate and enforce p.*

An evaluation authority can be explicitly defined if the policy writer directly knows and trusts it, or can be defined by characterization if the policy writer only knows which characteristics an entity must present to be trusted. Let DEA_p be the set of directly trusted evaluation authorities for a policy p and EA_p contain the set of directly trusted authorities and those defined by characterization. In the following we will use the terms *ea* and *TA* interchangeably. PAES' basic idea can be easily conveyed by adapting definition 2 to definition 1:

Definition 3. *Let p_i and p_j be two policies such that $p_j = (p_i, eval, c)$, where eval represents the policy evaluation action. Let ea be an entity acting in the system where p_i and p_j are defined. Then $ea \in EA_{p_i} \iff ea \in DEA_{p_i} \vee ea \in R_{p_j}$.*

The above definitions allow us to introduce the new concept of *policy chain*:

Definition 4. *A policy chain is a sequence of n policies $p_1 \rightarrow \ldots \rightarrow p_n$ such that $\forall p_i | i = 2 \ldots n : p_i = (p_{i-1}, eval, c)$.*

Thus each policy p_i ($i \geq 2$) is an *authority policy* that specifies the requirements an entity must satisfy to be considered an evaluation authority for policy p_{i-1} and p_1 is the *access control policy* for the data. Policy p_n, i.e. the last policy in the set, is always evaluated by an authority directly trusted by the policy writer (i.e. $EA_{p_n} = DEA_{p_n}$). Figure 2 illustrates the general format of a policy chain and the corresponding sets of evaluation authorities.

According to these definitions, an entity e is an evaluation authority that can vouch for the satisfaction of policy p_i by other entities (i.e. $e \in EA_{p_i}$) , if (i) it satisfies p_{i+1} evaluated by an evaluation authority $ea_j \in EA_{p_{i+1}}$, or (ii) the policy writer directly entitled it, i.e. $e \in DEA_{p_i}$.

When applying policy chains to existing rights management systems the set of authorities is no longer static but automatically changes when entities no longer satisfy the specified requirements or when new ones that do appear. Users (or evaluation authorities) that must be evaluated against a policy can choose their trusted evaluator among those satisfying the higher level policy and those being explicitly listed by the policy

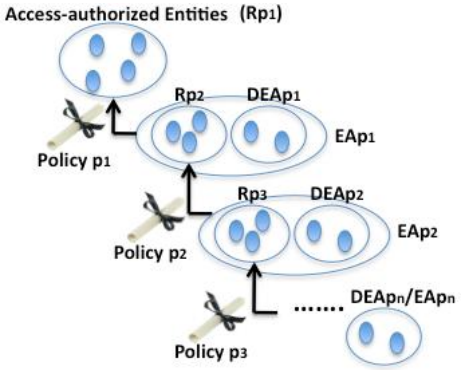


Fig. 2. Policy-defined groups of authorities

writer. They can for example choose the authority they trust the most with their confidential credentials. Moreover, evaluation chains can be built over different paths (i.e. with different authorities). If an authority goes offline or cannot be reached it can be easily replaced by another one satisfying the same requirements. This partly solves network problems, even if it still does not allow recipients to access the disseminated data if their devices are offline or cannot reach any authority.

With respect to trust management schemes such as TPL or RT, the trust relationship linking the policy writer to the data recipient still exists but in a restricted form. Data originators decide upon the policies composing the chain thus effectively controlling the delegation. For example, TPL allows specifying the trusted credentials but not the criteria according to which credentials are issued and who (apart from the policy writer himself) can evaluate if the criteria are met. We discuss other advantages of PAES in the next section where we also present a possible implementation of the scheme.

Figure 3 shows an example chain from the scenario described in Section 3. Each level specifies the entities authorised to evaluate the previous policy, either as an explicit list, a higher-level policy, or both. The figure shows a possible SHH corporate policy governing access to patients' records specifying that only MRC biochemists specialised in stem cells research can access them. SHH lets each patient agree with the policy and choose a set of entities he trusts to evaluate it. The patient considered in the example chooses Bob, one of his closest friends, to determine whether recipients satisfy the requirement. If Bob is a journalist most researchers would not want him to know they work on such a delicate research area. However, the policy also allows biochemists to be evaluated by any research facility working on stem cells research ranked by USNews as one of the best hospitals. Evaluation of whether a research facility satisfies this policy is left to SHH. SHH and the research facility may however be competitors, and the facility may not trust SHH to know what kind of research projects it is carrying out. Therefore the policy also allows research facilities to be evaluated by any publicly recognised organisation giving grants for medical research. Whether a company or organisation satisfies this requirement is directly left to verify to the European Union Research Commission (EURC). Note that SHH and patients can define authorities according to their liking and that they do not need to know in details the structures of the organisations involved but only sufficiently to state their requirements.

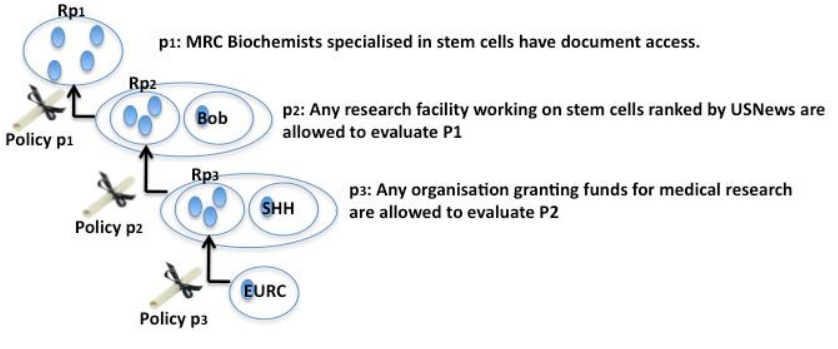


Fig. 3. Policy chain example

In the following section we describe a possible implementation of a data protection mechanism and protocol based on PAES that permits the evaluation and enforcing of policy chains.

5 A Possible Implementation

To implement the principles described above we propose a three phase protocol comprising: (i) the policy chain definition, (ii) the data protection and (iii) the policy chain evaluation. The first two phases concern protecting the resource from being accessed by non-authorised subjects. The third phase concerns policy evaluation and enforcement after the data has been received by recipients. The protocol is intended to be integrated with existing ERM systems.

Before describing the various phases, we introduce the basic notations and definitions necessary to formalise the protocol. Let E = $\{e_1, \ldots, e_e\}$ be the entities exchanging and accessing data and/or evaluating policies. Let P = $\{p_1, \ldots, p_p\}$ be the set of all possible policies. A policy evaluation is a boolean function $eval : E \times E \times P \rightarrow \{true, false\}$ such that $eval(e_l, e_m, p_i)$ denotes that e_m satisfies (or does not satisfy) the conditions imposed by p_i according to the evaluation made by e_l.

K denotes a symmetric encryption key and $\{D\}_k$ the data D encrypted with k. PK_e and PK_e^{-1} denote the private and the public key of an entity $e \in E$, and ID_e e's public key identity certificate (as in any PKI).

5.1 Policy Chain Definition Phase

At first, the data originator specifies a chain of policies $p_1 \rightarrow \ldots \rightarrow p_n$ as defined in Section 4 and an ordered list of sets of entities $\{DEA_{p_1}, \ldots, DEA_{p_n}\}$ such that each entity $ea_{ij} \in DEA_{p_i}$ is directly trusted to evaluate p_i. The specified policy chain is combined with the list to obtain a set of several *policy levels* $1, \ldots, n$ (with $n \geq 1$) where each level i defines a pair (p_i, DEA_{p_i}).

5.2 Data Protection Phase

In the *data protection phase* the data originator generates a symmetric key k_i for each specified level i except level n (the root of the chain is always a set of directly trusted evaluation authorities whose public keys are known by the originator) and a further key k_0. Using these keys the data D is encrypted as described below. The result of the protection phase is a concatenation of message parts $m_0, m_1, \ldots, m_n$ where:

- $m_0 = \{D\}_{k_0}$;
- $m_i = p_i | DEA_{p_i} | \{info_i\}_{k_i} | \{info_i\}_{PK_{ea_{i1}}} | \ldots | \{info_i\}_{PK_{ea_{id}}}$, where d is the cardinality of DEA_{p_i}, $info_i = (H(p_i), k_{i-1})$ for $1 \leq i \leq n$. $H(p_i)$ denotes a secure hash function applied to policy p_i;
- $m_n = p_n | DEA_{p_n} | \{info_n\}_{PK_{ea_{n1}}} | \ldots | \{info_n\}_{PK_{ea_{nd}}}$.

For simplicity we used the DEA_{p_i} to represent the sets of directly trusted authorities. In the actual implementation (see algorithm 1) for each entity $e \in DEA_{p_i}$ we include

the certificate ID_e in message m_i (although for the purpose of the protocol, it would be sufficient to represent each entity with a public key and a URL). In the data package shown above, the first symmetric level key k_0 is used to protect the data while each subsequent key k_i is used to protect k_{i-1} (k_{n-1} is only protected with the public keys of the directly trusted evaluation authorities at level n). A copy of each symmetric level key k_{i-1} is also protected with the public keys of the directly trusted evaluation authorities for p_i. Therefore, to obtain the data protection key k_0 each entity at level i must be authorised by an authority ea_{i+1}. At the root of the chain there is a directly trusted evaluation authority. Hashes of policies are included at each level to ensure policy integrity. Algorithm 1 describes the procedure to protect data according to a defined policy chain.

Input: Data D to be protected with an n-policy chain
$m_0 = \{D\}_{k_0}$;
for *i=1...n-1* **do**
  $info_i = (H(p_i), k_{i-1})$;
  $m_i = p_i | \{info_i\}_{k_i}$;
  foreach $ea_{ij} \in DEA_i$ **do**
    $m_i = m_i \,|\{info_i\}_{PK_{ea_{ij}}} | ID_{ea_{ij}}$;
  end
end
$info_n = (H(p_n), k_{n-1})$;
$m_n = m_n \,| p_n$;
foreach $ea_{nj} \in EA_n$ **do**
  $m_n = m_n \,|\{info_n\}_{PK_{ea_{nj}}} | ID_{ea_{nj}}$;
end
Output: $m_0 \ldots m_i \ldots m_n$

Algorithm 1. Data protection algorithm

The output of the data protection phase for the SHH patient's record in our example would then be:

m_0 $\{Record\}_{k_0}$ |
m_1 p_1(MRC biochemist working on stem cells) | $\{H(p_1), k_0\}_{k_1}$ | $\{H(p_1), k_0\}_{PK_{Bob}}$ | ID_{Bob}
m_2 p_2(Stem cells facility with USnews rank ≤ 10) | $\{H(p_2), k_1\}_{k_2}$, $\{H(p_2), k_1\}_{PK_{SHH}}$ | ID_{SHH}
m_3 p_3(Recognised org. granting funds) | $\{H(p_3), k_2\}_{PK_{EURC}}$ | ID_{EURC}

5.3 Policy Chain Evaluation Phase

The *policy chain evaluation* is a recursive procedure comprising an initial forward process (*policy evaluation*) and a final backward process (*key disclosure*). The policy evaluation finds a chain of entities satisfying the policies at each level. The initiator is the data recipient e_0 that receives the data with the messages $m_0, m_1, \ldots, m_n$ and tries to access D. To do so it first looks for an entity $ea_{1j} \in DEA_{p_1}$ he trusts and that grants him access under p_1. If the search succeeds e_0 sends m_1 to ea_{1j} (in practice: fields $\{info_i\}_{PK_{ea_{ih}}}$ for different authorities ($h \neq j$) are no longer useful and removed). The evaluation authority ea_{1j} can then decrypt and return k_0 with which e_0 can access D. Otherwise, e_0 can choose any entity $e_1 \in E$ he trusts and that grants him access under p_1. Then e_0 removes message m_0 and sends the rest of the messages to e_1. To return k_0, e_1 must first access

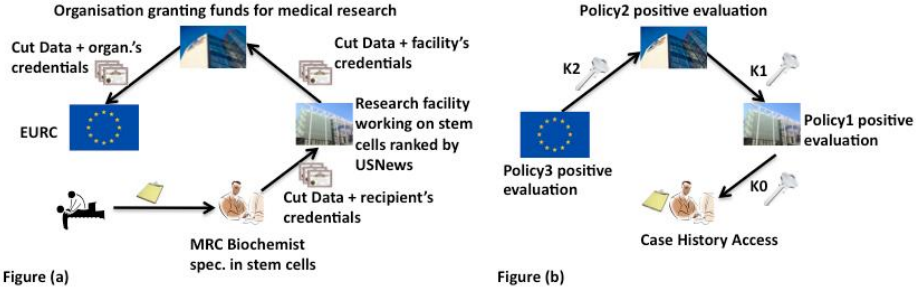


Fig. 4. Policy evaluation recursion

k_1 and therefore iterates the process (i.e. it looks for an entity that positively evaluates it under p_2). The sequence of evaluations terminates whenever an entity $ea_{ij} \in DEA_{p_i}$ returns a positive evaluation. To summarise, the forward iteration process finds a chain of entities $e_0, \ldots, e_h$ (if it exists) s.t. : (i) $1 \leq h \leq n$, (ii) $eval(e_{i+1}, e_i, p_{i+1}) = true$, (iii) e_i trusts e_{i+1} $\forall\, 0 \leq i < h$, and (iv) $e_h \in DEA_h$.

In the key disclosure process each entity e_i (where $i \neq n$) satisfying p_{i+1} is added to the set of evaluation authorities EA_i for policy p_i as it satisfied the criteria to be authorised to evaluate p_i (i.e. p_{i+1}). This is done by allowing each entity to get the symmetric level key necessary to disclose the information at the preceding level (i.e. k_{i-1}). This process is initiated by the last entity in the chain e_h. Entity e_h can directly access the information $info_h = H(p_h), k_{h-1}$ encrypted with its public key. It can then send k_{h-1} to entity e_{h-1} that will disclose k_{h-2} iterating the process. Each entity e_i in the chain sends to entity e_{i-1} the key k_{i-1} to decrypt the information $info_{i-1}$ until $info_0$, i.e. the data, is decrypted. Note that at each backward iteration the hash of each policy p_i is checked.

Figure 4 shows the execution of the two phases in our example. The recipient of a patient's record removes m_0 and sends the data, along with his credentials (certifying that he is a biochemist working for MRC and specialised in stem cells) to a research facility. Since policies are public (see Section 7 for motivation), he should look for a facility running a project on stem cells and ranked by USNews, as specified in the requirements. We call this entity e_1. e_1 verifies that the recipient is a MRC biochemist specialised in stem cells, removes message m_1 and sends the data, along with its credentials, to an organisation that funds research projects. We call this entity e_2. e_2 verifies that e_1 is a facility working on stem cells and ranked by USNews, removes message m_2 and sends the data, along with its credentials, to the European Union Research Commission (e_3). The EURC can directly access $h(p_3)$ and k_2 and verify that e_2 is a company whose research grants are publicly recognised. This concludes the forward process.

During the backward process e_3 sends k_2 to e_2, who can now access and verify the integrity of p_2. If the policy was not tampered with it can send k_1 back to e_1 that can now access and verify the integrity of p_1. If the policy was not tampered with it can send k_0 back to the data recipient that can now access the document.

Each entity trusts the one at the next level to see its credentials and to correctly verify if it satisfies the requirements specified in the policy. Trust however is not transitive since it is strictly related to the evaluation of a policy which involves only two nodes.

Credentials sent from an entity to an authority to be evaluated are in fact not further disseminated and confidential information is accessed only by authorities trusted to do so. Finally, since at each step a part of the data package is removed, entities at each level can only access the information they have been authorised for. With respect to trust management systems, recipients no longer need to gather credentials from every entity involved in the chain. At each step two different entities (evaluator and evaluated entity) perform a part of the protocol and evaluate part of the policy chain. At the end, the data recipient only has to present his own credentials to the chosen authority and does not worry about the other entities being part of the chain.

6 A Policy Language for Policy Chains

We propose a language that can be used to express policy chains, i.e. to combine basic policies in a form that can be interpreted during the PAES data protection phase. Chains expressed in this language convey the same semantic for policies and policy chains as that described in Section 4 but in a more readable syntax. Although the syntax is loosely based on the SecPAL language, the evaluation is performed according to the protocol described in the previous section. A PAES policy chain is specified as a sequence of policies, where each policy is the combination of a *permission* and a *policy statement*:

Subject **says** permission **if** policy statement

A permission represents the assignment of a right to a subject in the form *subject can do action* while a *policy statement* represents the corresponding conditions. Note that the policy-chain language is agnostic to the syntax used to specify policy statements. Any usage control policy language could be used provided that evaluation authorities can interpret it. We express the authorisation to access data as:

PolicyWriter **says** x **can** access data **if** $P_1(x, data)$

where $P_1(x, data)$ is a policy that grants x access to the data. Authority policies are expressed as:

PolicyWriter **says** y **can** say $P_1(x, data)$ **if** $P_2(y, P_1)$

where $P_2(y, P_1)$ is a policy that grants y the right to verify if x satisfies policy P_1. In contrast with SecPAL, note that delegated evaluation rights cannot be delegated further. A policy chain can be simply represented as:

PolicyWriter **says** x **can** access data **if** $P_1(x, data)$,
PolicyWriter **says** x **can** say $P_1(y, data)$ **if** $P_2(x, P_1)$, PolicyWriter **says** ea_{1j} *cansay* $P_1(y, data)$..
PolicyWriter **says** x **can** say $P_2(y, P_1)$ **if** $P_3(x, P_2)$, PolicyWriter **says** ea_{2j} *cansay* $P_2(y, P_1)$..
.....
PolicyWriter **says** $ea_n j$ **can** say $P_n(y, P_{n-1})$..

where the terms $ea_{1j}...ea_{nj}$ are used to indicate the evaluation authorities directly trusted by the policy writer. Using a hypothetical language for policy statements, we can now represent the policy chain and set of directly trusted authorities for our example as:

SHH **says** x **can** access case-history
if (x.role = Biochemist AND x.org = MRC AND x.spec = stem-cells),
SHH **says** Bob **can** say (y.role = Biochemist AND y.org = MRC AND y.spec = stem-cells)
SHH **says** x **can** say (y.role = Biochemist AND y.org = MRC AND y.spec = stem-cells)
if (x.facility = Research AND x.activity = stem-cells AND x.USNewsrank $\leq$ 10),
SHH **says** SHH **can** say (y.facility = Research AND y.activity = stem-cells AND y. USNewsrank $\leq$ 10),
SHH **says** x **can** say (y.facility = Research AND y.activity = stem-cells AND y. USNewsrank $\leq$ 10) **if** (x.role = PublicResFunder),
SHH **says** EURC **can** say (y.role = PublicResFunder)

Note that the example does not specify which entities must issue the required credentials (e.g. the public recognition as research funder being issued by the United Nations Institute for Training and Research, UNITAR, or USNews signing a rank certificate). This is typically addressed in the language adopted for policy statements.

7 Discussion

PAES represents a generalisation of the traditional ERM protocol. The originator (or an authority he trusts) could in fact be included as first directly trusted authority in the chain or second for collaboration between two domains (the first one being the authority in the partner domain), thus implementing the traditional ERM protocol.

Limitations. A limitation of PAES is that when one of the policies is evaluated negatively, the entity under evaluation must choose a different evaluator or return back to the previous entity a negative response. This also happens if no trusted evaluators can be found. Therefore, a policy chain evaluation may return a negative result even if the entity under evaluation actually satisfies the specified requirements. The flexibility that PAES offers to receive a negative response and try a different evaluation authority also raises a scalability issue. If n entities are required to evaluate a policy at each level of a chain spanning l levels, in the worst case the number of messages exchanged would be $O(n^l)$. However, we must consider that: i) an evaluation chain can be shorter than the corresponding policy chain (e.g. if the chosen evaluation authority is directly trusted by the policy writer), ii) policy recipients can filter out the entities they do not trust or that do not probably satisfy the chain requirements, iii) policy chains for most application scenarios are very short. Note that PAES allows the originator to specify his/her own policies for ascertaining his/her trust at each level. Leaving the policy specification to the data originator places a burden on him/her, but in practice many users are likely to use similar policies. Furthermore, as our example shows, corporate or regulatory policies can be directly used while the originator can simply integrate them with "shortcuts", i.e. with the specification of authorities he trusts to perform the evaluations.

Although the PAES approach offers greater flexibility, data recipients are still limited in their choice of evaluation authorities to those acceptable to the originator as defined in the policies. A negotiation process between data recipients and originators could offer more flexibility but would not be feasible in a data dissemination environment where users may not know each other and may be online at different times. Moreover, this may require the data to be repackaged for each negotiating recipient and would result

in a set of disseminated packages containing the same information but being controlled by different rules.

Possible Attacks. PAES aims to protect the confidentiality of disseminated data but remains prone to an attack that does not affect data confidentiality but the system availability, i.e. a denial-of-service attack (DOS). PAES policies are attached to the data as clear text so that they can be evaluated before the data is sent to the upper level. This is done to avoid the construction of a chain that may fail in the backward process. Only policies' hashes are protected to verify policy integrity. If a policy is corrupted then its chain can be destroyed (as well as its copies already disseminated) and the resource consumption would correspond only to the cost of an evaluation chain construction. No protected information would be disclosed to non-authorised entities. However, an attacker may disseminate many tampered policy chains in the network, thus causing the consumption of a large amount of resources (an evaluation chain must be constructed to discover every tampered package). The best solution to mitigate DOS attacks is that of identifying the packages' sources. This can be done for example by having data originators sign all the policies in the chain. Therefore authorities could verify the signature before executing the protocol and, if the policies have been tampered with, they could simply include the originator's key in a blacklist or revocation list.

Applicability. PAES may have different applications. Our work was motivated by ERM systems where our solution allows data originators to define who can access the data before the dissemination, without necessarily knowing the result of the successive access attempts and the set of possible recipients. However, the same protocol could also be used for data dissemination in Peer-To-Peer and ad-hoc networks. Every peer could define the requirements recipients must satisfy to access the data and how the satisfaction of these requirements must be evaluated by other trusted peers. In this sense, a particular application of the protocol could be represented by data exchange through e-mails. In this case PGP (Pretty Good Privacy) keys could be used.

An other interesting application is that of privacy protection. Consider a user accessing a remote service that requires a form to be filled with his private information. Assuming the company issuing the service is willing to provide such protection (e.g. to attract more clients) it could let the user define his own protection policies for the data inserted in the form and protect it before it is sent to the server. This would allow him to specify that data can be accessed only under the national regulations as evaluated by a known and trusted independent authority.

8 Conclusions and Future Work

The fact that different organisations can work on joint projects does not imply that the single individuals and entities of each party directly trust each other. The same can be said for entities working for the same organisation. Existing ERM systems and more generally policy evaluation mechanisms assume instead the existence of such trust relationships and force users to disclose their credentials to unknown entities. PAES fills this gap by allowing data recipients to choose their own trusted policy evaluators among those satisfying criteria defined by the data originator. Also, it enhances existing solutions

by distributing the policy evaluation process, so that the set of evaluation authorities can dynamically change. This is useful whenever some authorities are unknown or unreachable. Finally, PAES implements the Sticky Policy paradigm allowing data originators to distribute data with no need of an online publication phase. We have thus shown that simple extensions to existing policy evaluation systems can bring improvements in terms of flexibility and resiliency of the evaluation process. However, these improvements also have a complexity cost in some cases.

Future work will focus on an evolution of the protocol to deal with threshold policies (i.e. policies that must be evaluated by more than one authority). We will also allow different parts of the same policy (e.g. connected by AND/OR operators) to be evaluated by authorities satisfying different requirements. This means realising policy trees rather than simple chains. We aim to further evaluate the use of signed credentials attesting policy chain evaluations and thus how PAES can be integrated with existing certification systems.

Acknowledgments

We acknowledge financial support from the EC Consequence project (Grant Agreement 214859). We are also grateful to our colleagues G. Russello, L. Mostarda and C. Dong for many useful discussions and constructive comments.

References

1. eXtensible Access Control markup language (xacml) (version 2.0, 2005), http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf
2. Liquid machines and microsoft windows rights management services (rms): End-to-end rights management for the enterprise (2006), http://www.cmdsolutions.com/pdfs/LiquidMachines%20Windows%20RMS%20Business%20White%20Paper%20FINAL%20060213.pdf
3. Ashley, P., Hada, G., Karjoth, S., Powers, C., Schunter, M.: The enterprise privacy authorization language (epal 1.1) - reader's guide to the documentation -. Technical Report 93951, IBM (2003)
4. Authentica. Enterprise rights management for document protection, White Paper (2005)
5. Becker, M., Fournet, C., Gordon, A.: Secpal: Design and semantics of a decentralized authorization language. Technical Report MSR-TR-2006-120, Microsoft (2006)
6. Becker, M., Fournet, C., Gordon, A.: Design and semantics of a decentralized authorization language. In: CSF 2007: Proc. 20th IEEE Computer Security Foundations Symposium, Washington, DC, USA, pp. 3–15 (2007)
7. Boneh, D., Franklin, M.: Identity based encryption from the weil pairing. In: Proc. 21st Annual Int. Cryptology Conference, Santa Barbara, USA, pp. 213–229 (2001)
8. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
9. Clarke, D., Elien, J.-e., Ellison, C., Fredette, M., Morcos, A., Rivest, R.L.: Certificate chain discovery in spki/sdsi. J. of Computer Security 9 (2001)
10. Felten, E.W.: Understanding trusted computing: Will its benefits outweigh its drawbacks? IEEE Security and Privacy 1(3), 60–62 (2003)

11. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proc. 13th ACM Conf. on Computer and Communications Security, pp. 89–98. ACM, New York (2006)
12. Content Guard. extensible rights markup language (xrml) 2.0 (2001), http://www.xrml.org/
13. Housley, R., Ford, W., Polk, W., Solo, D.: Internet x.509 public key infra'structure certificate and crl profile. Request for Comments: 2459 (1999), www.ietf.org/rfc/rfc2459.txt
14. Iannella, R.: Open digital rights language (odrl), version 1.1. W3c note, World Wide Web Consortium (2002), http://www.w3.org/TR/odrl
15. Li, N., Mitchell, J.-C., Winsborough, W.H.: Design of a role-based trust management framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 114–130. IEEE Computer Society Press, Los Alamitos (2002)
16. Microsoft. Technical overview of windows rights management services for windows server 2003. White Paper (2005), download.microsoft.com/download/8/d/9/8d9dbf4a-3b0d-4ea1-905b-92c57086910b/RMSTechOverview.doc
17. Mont, M.-C., Pearson, S., Bramhall, P.: Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In: DEXA Workshops, pp. 377–382 (2003)
18. Park, J., Sandhu, R.S., Schifalacqua, J.: Security architectures for controlled digital information dissemination. In: 16th An. Computer Security Applications Conf. (ACSAC), New Orleans, USA, p. 224. IEEE Computer Society, Los Alamitos (2000)
19. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
20. Sandhu, R.S., Ranganathan, K., Zhang, X.: Secure information sharing enabled by trusted computing and pei models. In: ASIACCS, pp. 2–12 (2006)
21. Sandhu, R.S., Zhang, X., Ranganathan, K., Covington, M.: Client-side access control enforcement using trusted computing and pei models. J. High Speed Networks 15(3), 229–245 (2006)
22. Schoen, S.: Trusted computing: Promise and risk (2003), http://www.eff.inorg/files/20031001_tc.pdf
23. Avoco Secure. Choosing an enterprise rights management system: Architectural approach (2007), www.windowsecurity.com/uplarticle/Authentication_and_Access_Control/ERM-architectural-approaches.pdf
24. Yu, Y., Chiueh, T.-c.: Enterprise digital rights management: Solutions against information theft by insiders. Research Proficiency Examination (RPE) report TR-169, Department of Computer Science, Stony Brook University (2004)

Emerging Trends in Health Care Delivery: Towards Collaborative Security for NIST RBAC

Solomon Berhe[1], Steven Demurjian[1], and Thomas Agresta[2]

[1] Department of Computer Science & Engineering, University of Connecticut, U-2155, 371 Fairfield Road, Storrs, CT, USA
steve@engr.uconn.edu, solomon.berhe@engr.uconn.edu
[2] Department of Family Medicine, University of Connecticut Health Center, 263 Farmington Avenue, Farmington, CT, USA
Agresta@nso1.uchc.edu

Abstract. In the next 10 years there will be rapid adoption of health information technology - electronic medical records by providers and personal health records by patients - linked via health information exchange. There is an emergent need to provide secure access to information spread across multiple repositories for health care providers (e.g., physicians, nurses, home health aides, etc.) who collaborate with one another across cyberspace to deliver patient care. Are available security models capable of supporting collaborative access where providers are simultaneously modifying a patient's medical record? To address this question, this paper details collaborative security extensions to NIST RBAC.

1 Introduction

In 1990, two seminal articles were published related to health care security [7,20]. In [7], privacy and confidentiality in medical information systems was explored. In [20], a detailed case study of mental health delivery from information and semantic perspectives was presented. In the nearly 20 years since their publication, has security kept up with the current and emergent needs of health care delivery in 2010 and beyond? Today, health information technology (HIT) systems are widespread, including: electronic medial record (EMR) to manage all of the health-related information for each patient; electronic prescribing (eRx) to write and transmit prescriptions to pharmacies, personal health records (PHR) that place health-related information directly into the hands of patients. All of these systems must adhere to stringent HIPAA regulations (Health Insurance Portability and Accountability Act of 1996) [16] in regards to the security, availability, transmission, and release of a patient's medical information. In the US, the approved stimulus bill (H.R.1) contains significant funding for HIT adoption including EMRs, PHRs, and health information exchange (HIE).

The movement to a massively linked health information network will be accompanied by dramatic changes in health care delivery, particularly in regards to the way that health care providers collaborative communicate and interact

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 283–290, 2009.

with one another (and patients and their families) [1,2]. Patients with chronic conditions move from provider to provider (e.g., from an internist to a cardiologist) and from location to location (from their home to a hospital to a rehab center and back home again). A patient's medical record differs from data stored in a database in that no data is ever deleted; the record continues to grow over time and can be distributed across multiple EMRs and PHRs. The key is to maintain a complete medical record, and to provide models of collaboration for providers as a patient moves among providers or locations [21].

For collaborative health care, patient privacy and confidentiality must be protected while promoting shared access of information by providers; role-based access control (RBAC) [10,17,19] is a starting point. However, is NIST RBAC [13] well suited to address emergent HIT and HIE needs? Historically, in NIST $RBAC_2$, constraints for separation of duty (SOD), mutual exclusion, and cardinality [3,8] focus on prevention of actions by restricting behavior. For health care, with an "ever-increasing" patient record composed of multiple connected objects in different locations, we are concerned with limiting access while promoting effective and timely treatment. For collaborative care on a single patient, each provider could simultaneously add notes, treatment recommendations, prescriptions, etc. In this context, providers are required to *collaborate on duty* rather than separate their actions. In the remainder of this paper: Section 2 details background and related work; Section 3 proposes extensions to NIST $RBAC_2$ for *collaboration of duty (COD)*; and, Section 4, offers concluding remarks.

2 Background and Related Work

In this section, we present a scenario of patient care based on current and emerging technologies that involves a *virtual chart* [11] which collects a patient's medical history from multiple sources using HIE for unified access against a combined view. Mr. J Smith is a 78 year old patient with known diabetes and a long history of smoking who presents to the emergency room (ER) with shortness of breath and wheezing for the first time. The ER staff initiates a request for his current health records via a secure messaging portal. It is noted that the patient also has a G6 PD deficiency and an allergy to Penicillin. On exam, the patient has findings consistent with emphysema and the possibility of a newly evolving cardiac condition, so a chest X-Ray, an EKG, and some laboratory studies are ordered. The chest X-Ray demonstrates some findings consistent with emphysema and a minimal amount of fluid at the lung bases. The ER physician and ER nurse immediately contact a cardiologist and a radiologist (in another building) to collaboratively review the X-Ray and EKG; based on this collaboration and other test results, the decision to admit the patient is made.

The patient is seen by a hospitalist physician who orders the antibiotic Bactrim and is contacted by the pharmacist prior to dispensation of the medication to discuss how that can cause hemolysis when used in patients with G6 PD deficiency so an alternate is chosen. The patient continues to get better. The hospitalist communicates directly with the primary care physician via a web-based technology on the

day of discharge to discuss follow-up and medication use. They agree that the patient would benefit from a visiting nurse twice weekly over the next 3 weeks. The discharge summary is sent automatically to the patient's primary care physician and the patient's medications on discharge are sent via a E-Prescribing portal that also updates the EMR in the primary care office as well as the patient's PHR. The patient monitors his vital signs and weight daily, recording these into his PHR, which sends a flag automatically to his primary care physician's office if he falls outside of the agreed upon parameters. The patient follows up with his primary care physician 2 weeks after discharge and is doing markedly better. While this scenario is futuristic, all of the indicated systems are available, often in isolation and with limited HIE. The example demonstrates the collaborations that optimally would take place among health care providers to view and modify a patient's medical record.

As to related work, there are many areas that have influenced our work. First, there have been research efforts addressing RBAC [3,8,10,17,19], collaboration models [1], and health care [11]. We have examined these efforts to insure that we can define a collaborative security model based on NIST RBAC and applicable to the health care and other domains. [14] presented the impact of legal patients' rights and their security requirements. Our work shares the common goal of protecting patients' medical records with the help of RBAC. In terms of access control and collaboration [21], a set of eight criteria critical in an collaborative environment are presented. We have begun to evaluate the degree that our COD extensions address these criteria. Another related effort includes a general design for secure collaboration [12], which will be examined more closely against our model since their effort impacts on both the security design process and enforcement. In [4], a model for dynamic trust negotiation for health care collaborations is presented, necessary for emergency situations where collaboration and the associated sharing of patients' medical records is paramount. We will need to consider dynamic collaborations as we expand our COD to include workflow. Lastly, in [9], a list of inter-professional and inter-organizational collaboration challenges in the health care domain are reviewed, along with a validation model. Generalizing and applying such a validation to our work on COD will be an important step in assessing our work.

3 Collaboration on Duty via Extended NIST $RBAC_2$

For our purposes, the general act of collaboration can be defined as: "Two or more users (each with their own role) each with their own set of allowable actions who are accessing (read and/or write) one or more objects for the same entity with the possibility that their access is constrained by time (when an action can occur) or order (defined sequence of the order of actions)." This section explores collaboration by: reviewing the NIST RBAC model and its limitations for health care; and, extending NIST $RBAC_2$ with *collaboration of duty (COD)* constraints illustrated using an example collaboration from Section 2.

The NIST reference models, $RBAC_0$ to $RBAC_3$[17] are an ideal starting point for security as related to the scenario in Section 2. $RBAC_0$ links the concepts of

roles and permissions (permission assignment) and users and roles (user assignment). $RBAC_1$ allows the definition of role hierarchies where privileges assigned to one role are available to other roles depending on the defined relationship. For example, an ER physician would be a a child of a physician role. $RBAC_2$ extends $RBAC_1$ with constraints [3], namely, *separation of duty (SOD)* and *mutual exclusion* [8]. For example, in the early stages of training, an intern may wish to write a prescription for a narcotic (e.g., oxycontin) but may require a resident physician to approve and write the prescription; this illustrates static SOD. While NIST RBAC can handle typical requirements, there are changes emerging. There is a movement towards the *medical home* where care is coordinated by one physician who reaches out and collaborates with a myriad of health care providers [5,6]; Section 2 illustrated such a scenario. As a result, NIST RBAC is limited in support of collaboration by: no direct support for the idea of collaborative access of multiple (user, role) pairs on one or more (objects, action) pairs; overlapping roles in health care made it difficult to establish a clear cut boundary between responsibilities and tasks; and, the assignment of permissions based only on role is not sufficient to clearly take into consideration contextual concerns such as an individual's schedule, the collaboration team, availability of collaborators, sequence of collaborators, etc.. One premise of our work is that these actions are part of an enhanced security model that considers collaboration and workflow as integral to achieving secure information usage.

Given these limitations, we introduce notation consistent with NIST RBAC [13] to serve as a basis for its extension with COD. To begin, we define: $\mathcal{U}$,$\mathcal{R}$, and $\mathcal{P}$ as sets of users, roles, and permissions, respectively; $\mathcal{O}$ and $\mathcal{A}$ as sets of all *objects* and *actions*, respectively, on which permissions are defined for roles and then assigned to users; and, for *assignment*, we define three sets:

- $\mathcal{P} \subseteq \mathcal{O} \times \mathcal{A}$ be a many-to-many *object-action assignment*,
- $\mathcal{PA} \subseteq \mathcal{R} \times \mathcal{P}$ be a many-to-many *role-permission assignment*, and
- $\mathcal{UA} \subseteq \mathcal{R} \times \mathcal{U}$ be a many-to-many *role-user assignment.*

Note that SOD, cardinality, and mutual exclusion constraints determine whether or not values in $\mathcal{PA}$ and $\mathcal{UA}$ are permitted. For the exmaple from Section 2, we use Mr. Smith's initial assessments and laboratory tests (X-Ray and EKG) by defining the sets $\mathcal{U}$,$\mathcal{R}$, $\mathcal{UA}$, $\mathcal{O}$, $\mathcal{A}$, and $\mathcal{P}$:

- $\mathcal{U}$ = {ERPhysician1, ERNurse1, Cardiologist1, Cardiologist2, Radiologist1, Radiologist2, Patient1}
- $\mathcal{R}$ = {Physician, Nurse, PhysicianSpecialist, Patient}
- $\mathcal{UA}$ = {(ERPhysician1, Physician), (ERNurse1, Nurse), (Patient1, Patient) (Cardiologist1, PhysicianSpecialist), (Cardiologist2, PhysicianSpecialist), (Radiologist1, PhysicianSpecialist), (Radiologist2, PhysicianSpecialist)}
- $\mathcal{O} = \{o^{VC}_{J.Smith}, o^{X-Ray}_{J.Smith}, o^{EKG}_{J.Smith}\}$
- $\mathcal{A}$ = {read, write}
- $\mathcal{P} = \{P_1 = (\text{read}, o^{VC}_{J.Smith}), P_2 = (\text{write}, o^{VC}_{J.Smith}), P_3 = (\text{read}, o^{X-Ray}_{J.Smith}), P_4 = (\text{write}, o^{X-Ray}_{J.Smith}), P_5 = (\text{read}, o^{EKG}_{J.Smith}), P_6 = (\text{write}, o^{EKG}_{J.Smith})\}$

Instead of names, we use a job title and a number to signify a person. We assume PhysicianSpecialist is a child of Phyician. Note VC stands for virtual chart.

Given these definitions, we propose *collaboration on duty (COD)* extensions to NIST RBAC. As shown in the lower right of Figure 3, these constraints impact the COD Teams (User/Role combinations) and define the allowable actions on objects with respect to defined permissions. There are four different *COD Constraints (CODC)*: lifetime (LT) which indicates the range (time period) for when a collaboration team is active; time-to-complete (TTC) the collaboration which represents the maximum duration of the collaboration; cardinality (CARD) which denotes a range (minimum, maximum) of individuals (each with a user/role combination) who must participate in the collaboration; and, attendance (ATT) which captures the participation of the team members for the collaboration. When establishing a collaboration, the collaboration must specifically select one user/role combination to constrain the actions and affected objects of that user to that role in a collaboration:

Definition 1. A *Collaboration Team*, $\mathcal{CT} \subseteq \mathcal{UA}$, is defined as a set of user/role combinations to indicate the user and his/her role for that particular collaboration.

Definition 2. The *Collaboration Permissions*, $\mathcal{CP} \subseteq \mathcal{PA}$, represent the involved permissions of the $\mathcal{CT}$ for the collaboration.

Note that Collaboration Permissions are a subset of the defined permissions ($\mathcal{PA}$) for all of the user/role combinations on the Collaboration Team. Each member of the Collaboration Team is constrained to a subset of those objects, actions, and permissions that are defined for his/her role.

In terms of time, two optional constraints control when a collaboration occurs, lifetime (LT) and time-to-complete (TTC). LT indicates the time range (start and end time) when the team $\mathcal{CT}$ is operational for a collaboration. TTC represents the duration of the collaboration once it begins. Note that if LT is defined, a collaboration cannot start before the start time or complete after the end time. If a TTC is also defined, the collaboration must start to allow the duration to finish prior to the end time as defined by LT.

Definition 3. A *CODC for Lifetime*, $CODC_{LT} = [Start-Time, End-Time]$.

Definition 4. A *CODC for Time-to-Complete*, $CODC_{TTC} = [Duration]$.

Note that the Duration must occur within the LT (if defined); otherwise, it represents the duration of the collaboration once it begins.

Cardinality (CARD) is an aggregation constraint for the definition of the minimum and maximum number of participants. When minimum equals maximum equals the size of the team, then all team members must participate. Note that it does not denote which members must participate.

Definition 5. A *CODC for Cardinality*, $CODC_{CARD} = [min, max]$, where min $\leq$ max and max $\leq \mid \mathcal{CT} \mid$.

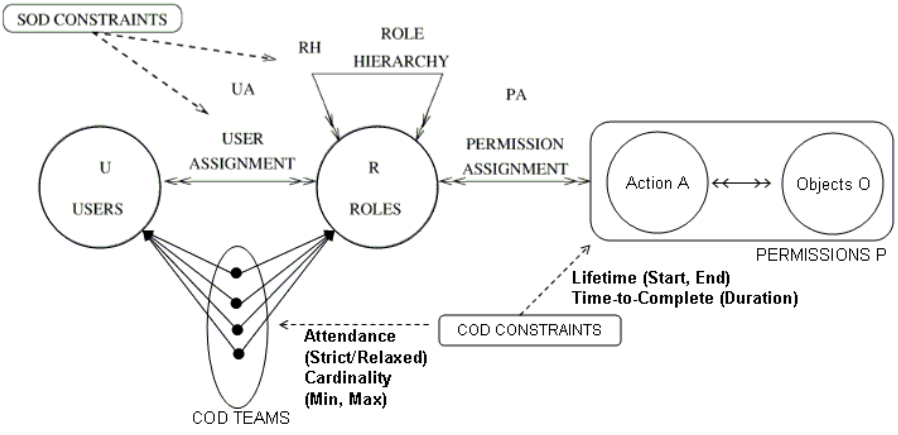


Fig. 1. COD Capable NIST $RBAC_2$

$CODC_{CARD}$ is not sufficient to dictate who must participate in a collaboration; we differentiate between the users in $\mathcal{CT}$ who must participate in the collaboration vs. those who may participate. When defining COCD attendance (ATT), we specify two sets of users: C_{strict} is a set of users who must participate in the collaboration, while $C_{relaxed}$ is a set of sets of users, where for each set of users, at least one must participate.

Definition 6. A *CODC for Attendance*, $CODC_{ATT} = \{C_{relaxed}, C_{strict}\}$ where:

- $C_{strict} \subseteq \mathcal{U}$ are the set of users who must participate in the collaboration,
- $C_{relaxed} = \{\{ua_1\}, \{ua_2\},...,\{ua_n\}\}$, where each ua_i ($\mid ua_i \mid \geq 2$), i ∈ 1..n, is a set of users, and for each set there exists at least one $u_j \in \mathcal{U}$ who participates in the collaboration, and
- $\forall_{i=1}^{n} ua_i (\in C_{relaxed}) : C_{strict} \cap ua_i = \emptyset$

Notationally, we collect all of the COD Constraints for a collaboration as:

Definition 7. The COD Constraints for a collaboration is:
$\mathcal{CODC} = [CODC_{LT}, CODC_{TTC}, CODC_{CARD}, CODC_{ATT}]$.

In this notation, any of the constraints may be null. Next, an individual collaboration brings together the collaboration team, permissions (a subset of the overall permissions $\mathcal{PA}$), and the constraints for that team as:

Definition 8. A Collaboration $\mathbb{C}$ is a three tuple defined as: $\mathbb{C} = (\mathcal{CT}, \mathcal{CP}, \mathcal{CODC})$.

For the example: $\mathbb{C}_1 = (\mathcal{CT}, \mathcal{CP}, \mathcal{CODC})$ where

- $\mathcal{CT}$ = {ERPhysician1, ERNurse1, Cardiologist1, Cardiologist2, Radiologist1, Radiologist2}

- $\mathcal{CP}$ = { (ERPhysician1,P_2, P_3, P_5), (ERNurse1,P_2, P_3, P_5), (Radiologist1,P_2, P_4), (Radiologist2,P_2, P_4), (Cardiologist1,P_2, P_3, P_6), (Cardiologist2,P_2, P_3, P_6) }
- $\mathcal{CODC}$ = $[CODC_{LT}, CODC_{TTC}, CODC_{CARD}, CODC_{ATT}]$ with
  - $CODC_{LT} = (\emptyset, \emptyset)$
  - $CODC_{TTC}$ = 1 hour
  - $CODC_{CARD} = (3, 4)$
  - $CODC_{ATT} = \{C_{relaxed}, C_{strict}\}$ where $C_{relaxed}$={{Radiologist1, Radiologist2}, {Cardiologist1, Cardiologist2}} and C_{strict}={ERPhysician1, ERNurse1}

$\mathbb{C}_1$ is a four-way collaboration between ERPhysician1, ERNurse1, a radiologist, and a cardiologist, with their permissions to read and write the patient's VC, X-Ray, and EKG as indicated by $\mathcal{CP}$. For COD constraints there is: a one hour time-to-complete ($CODC_{TTC}$); a requirement that at least one radiologist and at least one cardiologist join the collaboration ($C_{relaxed}$); and, a requirement that ERPhysician1 and ERNurse1 must attend (C_{strict}). Note that we have slightly eased the cardinality of the collaboration since according to busy schedules or other emergencies, we only require that 3 out of 4 individuals be present, but all four must participate at some point in the time limit.

Finally, a given application consists of multiple collaborations:

Definition 9. An Applications (APP) Collaborations $\mathcal{APP}_{\mathbb{C}} = \{\mathbb{C}_j\}$ for j=1..m collaborations.

This definition captures collaborations for the entire application, to be complemented with roles, users, permissions, SOD, mutual exclusion, etc.

4 Conclusion

In this paper, we have revisited two classic articles [7,20], and used them as rationale to look forward to understand the rapidly changing and evolving role of information technology in health care. Electronic medical records, personal health records, electronic prescribing systems, with health information exchange to tie them all together will offer new opportunities for health care professionals and providers to collaborate with one another towards improved patient care. This paper has focused on the extension of NIST RBAC to support collaboration of duty (COD) by: presenting a futuristic health care scenario collaboration in Section 2; and, COD extensions to $RBAC_2$ to capture collaborations and constraints with respect to time, access, and attendance in Section 3, supplemented with an illustrative example from the futuristic scenario. We believe that the work presented herein is an important step in introducing collaboration into the security and NIST RBAC.

References

1. Abraham, J., Reddy, M.: Moving Patients Around: A Field Study of Coord. Between Clinical and Non-Clinical Staff in Hospitals. In: Proc. of ACM 2008 Conf. on Computer Supported Cooperative Work (2008)
2. Agrawal, R., et al.: Enabling the 21st Century Health Care Information Technology Revolution. Comm. of the ACM
3. Ahn, G.-J., Sandhu, R.: Role-Based Authorization Constraints Specification. ACM Trans. Inf. Syst. Secur. 3(4) (2000)
4. Ajayi, O., et al.: Dynamic Trust Negotiation for Flexible E-Health Collaborations. In: Proc. of 15th ACM Mardi Gras Conf. (2008)
5. American Academy of Pediatrics Web Page and Discussion on Medical Home, `http://www.medicalhomeinfo.org/`
6. American College of Physicians Web Page and Discussion on Medical Home, `http://www.acponline.org/advocacy/where_we_stand/medical_home/`
7. Biskup, J.: Protection of Privacy and Confidentiality in Medical Information Systems: Problems and Guidelines. In: Spooner, D., Landwehr, C. (eds.) Database Security, III: Status and Prospects. North-Holland, Amsterdam (1990)
8. Chen, H., Li, N.: Constraint Generation for Separation of Duty. In: Proc. of 11th ACM Symp. on Access Control Models and Technologies (2006)
9. D'Amour, D., et al.: A Model and Typology of Collaboration Between Professionals in Healthcare Organizations. BMC Health Services Research (2008)
10. Ferraiolo, D., et al.: Proposed NIST Standard for Role-Based Access Control. ACM Trans. on Information and Sys. Sec. 4(3) (2001)
11. Kenny, P., et al.: Virtual Humans for Assisted Health Care. In: Proc. of 1st Intl. Conf. on PErvasive Technologies Related to Assistive Environments (2008)
12. Nakae, M., et al.: A General Design Towards Secure Ad-hoc Collaboration. In: Proc. of 2006 Symp. on Information, Computer and Communications Security (2006)
13. NIST RBAC Standard, `http://csrc.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf`
14. Ali Pabrai, U.O.: Getting Started with HIPAA. Course Technology Press (2003)
15. Park, J., et al.: A Secure Workflow System for Dynamic Collaboration. In: Sec 2001: Proc. of 16th Intl. Conf. on Information Security: Trusted Information (2001)
16. Rindfleisch, T.: Privacy, Information Technology, and Health Care. J. of the ACM 40(8) (1997)
17. Sandhu, R., et al.: Role-Based Access Control Models. IEEE Computer 29(2) (1996)
18. Sims, S., et al.: Surveillance of Methadone-Related Adverse Drug Events Using Multiple Public Health Data Sources. J. of Biomedical Informatics 40(4) (2007)
19. Ting, T.C.: A User-Role Based Data Security Approach. In: Landwehr, C. (ed.) Database Security: Status and Prospects. North-Holland, Amsterdam (1988)
20. Ting, T.C.: Application Information Security Semantics: A Case of Mental Health Delivery. In: Spooner, D., Landwehr, C. (eds.) Database Security, III: Status and Prospects. North-Holland, Amsterdam (1990)
21. Tolone, W., et al.: Access Control in Collaborative Systems. ACM Computing Surveys 37(1) (2005)
22. Xiao, Y.: Artifacts and Collaborative Work in Healthcare: Methodological, Theoretical, and Technological Implications of the Tangible. J. of Biomedical Informatics 38(1) (2004)

Methods for Computing Trust and Reputation While Preserving Privacy

Ehud Gudes, Nurit Gal-Oz, and Alon Grubshtein

Dept. of Computer Science and Deutsche Telekom Labs at Ben-Gurion University, Beer-Sheva, 84105, Israel
ehud@cs.bgu.ac.il, galoz@cs.bgu.ac.il, alongrub@cs.bgu.ac.il

Abstract. Trust and Reputation systems in distributed environments attain widespread interest as online communities are becoming an inherent part of the daily routine of Internet users. Trust-based models enable safer operation within communities to which information exchange and peer to peer interaction are centric. Several models for trust based reputation have been suggested recently, among them the Knots model [5]. In these models, the subjective reputation of a member is computed using information provided by a set of members trusted by the latter. The present paper discusses the computation of reputation in such models, while preserving members' private information. Three different schemes for the private computation of reputation are presented, and the advantages and disadvantages in terms of privacy and communication overhead are analyzed.

1 Introduction

Recent years have seen a substantial growth of virtual communities across the Internet. The accessibility of information and services offered by these communities, makes it both possible and legitimate to communicate with strangers and carry out interactions anonymously, as rarely done in "real" life. However, virtual communities are prone to many types of deception ranging from people forging their identity and imposing as others, to people rating their service providers extremely high or extremely low unrelated to the service they have received. Trust and Reputation systems (TRS) provide communities with means to reduce the potential risk when communicating with people hiding behind virtual identities. These utilize the experience and knowledge accumulated and shared by all participants for assigning reputation values to individuals, and attempt to identify dishonest members and prevent their negative effect.

Reputation may depend on factors such as: interaction of multiple attributes (as in eBay), certainty of ratings [7], time of interaction and rating, and trust between members. The latter is crucial in obtaining reputation which is specifically compatible with a user profile or preferences. The issue of trust between members is discussed in [3,5] among others. When anonymity is required, trust between members may be computed based on the similarity of their past ratings (see [5]). As more systems implement a trust mechanism between members the

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 291–298, 2009.

impact of this information grows, and greater efforts should be made in keeping it private.

An empirical study conducted by [12] on data sets from eBay's reputation system reported a high correlation between buyer/seller ratings. Moreover, most feedback provided was positive. One explanation for these results is that when feedback providers' identities (pseudo-identities) are known, ratings are provided based on reasons of reciprocation and retaliation, not properly reflecting the trustworthiness of the rated parties. Thus preserving the trust level of members in each other private, while computing reputation becomes an important issue.

Decentralized reputation systems do not make use of a central repository to process reputation ratings [15] and are considered safer. In these, both the reputation of users and the ratings they give may be stored locally and known only to the corresponding user. The challenge here is to compute the reputation while maintaining private data. Its important to emphasize that based on the TRS, there exist different data which may be considered private, such as ratings, weights assigned to specific ratings, identity of the raters, trust between members, etc. A recent paper [9] suggested several privacy preserving schemes for simple computation of reputation (e.g. eBay [1]).

The current paper advances the state of the art as it uses an enhanced model of reputation computation which considers the trust members have in one another. We base our discussion on the *Knots* model [5] but the ideas presented may apply to any model in which trust between members is important in computing reputation. *Three* different methods for computing trust and reputation privately are presented, each offering a slightly different degree of privacy and communication overhead. All are analyzed with respect to users' malicious and non-malicious behavior, and the advantages of each are discussed. The rest of the paper is organized as follows. Section 2 provides an overview of related work and presents the knots model. Section 3 describes the three schemes and in Section 4 we conclude and give some future research directions.

2 Background and Related Work

The concern for privacy in communities in general, and the privacy of reputation information in particular, was discussed in several papers [8,6,14,11,4]. In [8] the authors discuss issues of privacy in a P2P network where reputation information is distributed among the P2P nodes. Requirements for fair use practices and their impact on system design, classes of information which may be leaked and managing the risks related to social and technical issues are all analyzed. However, no specific method for computing reputation is presented. In [6] a distributed trust management system, constructed from a two level hierarchy is described. The high level is composed of brokers which are responsible for aggregating trust information from their individual local nodes. This provides some privacy from one broker to another, although no privacy is provided at a single broker's network. Steinbrecher in [14] presents an information theoretic model of reputation privacy. She tries to model the amount of privacy lost when a single

user uses different pseudonyms in the same community or in different ones, or when a user changes her pseudonym. Her measure enables the estimate of *unlinkability* provided by such a pseudonym change. In [11], the authors discuss the issue of privacy for a user in multiple communities that requires a transfer of the reputation between these communities, creating the concept of cross-community reputation (CCR). CCR requirements were analyzed in [4] including issues of privacy, user control vs. community control, ontology matching and others.

Pavlov et al [9] provide the closest paper to the present work. It deals with private computation of reputation information, when the reputation is defined as an additive reputation system (e.g. the Beta reputation system [7]). The authors present three algorithms for computing additive reputation, with various degrees of privacy and with different level of protection against malicious users. A method for "witness selection" which reduces the risk of selecting dishonest witnesses is first presented. Then, a simple scheme which is very efficient but vulnerable to collusion of even two witnesses is introduced. The second scheme is more resilient towards curious users, but vulnerable to collusions and uses a secret splitting scheme. The last, provides the most secure protocol which uses the verifiable secret sharing scheme [10] based on Shamir's secret sharing scheme [13]. The complexity of the scheme is quite high and requires $O(n^3)$ messages where n is the number of contributing nodes. These schemes can be used with additive computations. Our work, which is described in terms of the Knots model [5] uses additional factor, which is not additive.

2.1 The Knots Model

The Knots model [5] is a TRS model for large-scale virtual communities. It is composed of three modules: *member trust inference module*, *Knots construction module*, and *reputation computation module*. The member trust inference module identifies trust relations among members; the Knots construction module utilizes these relations to generate trust Knots; the reputation computation module computes local reputations within Knots and global reputations for the whole community. Without loss of generality we adopt the notation from this model, but the underlying approach exists in other models as well.

- *TrustMember* (TM) is trust in the context of recommendations. It is a trust value that quantifies the extent by which one member relies on another member to rate experts "correctly".
- *TrustExpert* (TE) is trust in the context of experts. Specifically, it quantifies the extent by which a member relies on an expert to successfully provide the service it requires.
- An α - *Trust Set* of a member A, is the set of all members whom A trusts with level α or more.

The motivation for our current paper is to provide trust based reputation models such as the knot model, with means to compute reputation in a distributed manner while preserving private information. In this context the trust one member has in another (TM) and the trust a member has in an expert (TE), are

both considered private information. One would prefer not to reveal her trust in a member from which she gets recommendations. On the other hand the recommending party may prefer to keep her recommendation private due to the fear of retaliation. Revealing these to malicious parties exposes the community to the risk of manipulating recommendations.

3 Privacy Preserving Trust and Reputation Computation

The problem of privacy we address is the following: assume the existence of an expert x, and a member A. A needs to compute her trust in the expert, based on the experience other members of the community have had with this expert. We use the Trust-set as the group of members participating in this computation. The trust of A in x using Trust-sets [5] can be computed according to:

$$TE(A,x) = \frac{\sum_{\substack{B_i \in TrustSet(A), \\ DTE(B_i,x) \neq \perp}} DTE(B_i,x) \cdot TM(A,B_i)}{\sum_{\substack{B_i \in TrustSet(A), \\ DTE(B_i,x) \neq \perp}} TM(A,B_i)}$$

where:

- $DTE(B_i,x)$ - the trust member B_i has in expert x based on her own accumulated experience.
- $ITE(A,B_i,x)=TM(A,B_i) \cdot DTE(B_i,x)$ - the indirect trust member A has in expert x based on B_i's direct trust in x.

We assume that $TM(A,B)$ is known to agent A since it reflects her private information. Therefore the denominator is easy to compute by A without disclosing private information. The nominator is a sum of products of two terms, the first one is assumed to be distributed among the agents B_i, and known only by its corresponding agent, and the second one is known to A. Therefore the challenge we face is to privately compute the following sum of products, denoted by $\rho(A,x)$:

$$\rho(A,x) = \sum_{\forall B_i \in S} DTE(B_i,x) \cdot TM(A,B_i) = \sum_{\forall B_i \in S} ITE(A,B_i,x)$$

where S denotes the trust-set of A.

We proceed to presenting three different schemes to compute this sum. In all three schemes we assume a *semi-honest* protocol. That is, members follow the protocol honestly, but may remember information and use it for their own benefit. In section 3.1, we discuss the case of dishonest users.

Scheme 1: Trust relations aware
The first scheme, presented in Algorithm 1, assumes that every member $B_i \in S$ knows the trust value, $TM(A,B_i)$, that member A has in her, or that A is willing to disclose this information to B_i. It is also assumed that a partially trusted third party Z exist (i.e. Z does not collude with other agents, and she has no access to any private information of the involved parties).

Algorithm 1. Trust relations aware

1: A sends around her public key K_A to all $B_i \in S$ along with $TM(A, B_i)$
2: Each member B_i computes $ITE(A, B_i, x)$, encrypts it and sends the encrypted form $C(K_A, ITE(A, B_i, x))$ to the third party Z.
3: Z generates a random permutation of the encrypted messages and sends it to A.
4: A decrypts the messages and computes the required sum $\rho(A, x)$.

The encryption ensures that only B_i knows $ITE(A, B_i, x)$. The permutation carried out by Z ensures that when the set of values $ITE(A, B_i, x)$ is received by A, the origin of the $ITE(A, B_i, x)$ value is not clear to A. This scheme achieves our computational goal and therefore enables private computation of reputation by any member A. Collusion between any of the members (excluding Z) in this scheme is not helpful, since they have no access to information related to non-colluding members. The main advantage of this scheme is its simplicity and small communication overhead (basically $O(n)$), while its main disadvantage is the disclosure of member trust values. Although deemed otherwise, this privacy violation is not very severe: being in A's trust-set, members already know that their corresponding trust value is above the threshold α.

Scheme 2: Trusted third party dependent
In this scheme, depicted in Algorithm 2, we try to overcome the major disadvantage of the former method and assume that members do not know the trust other members have in them. Unlike our previous scheme, only a trusted third party Z is entrusted with the values of the trust A has in the different members, i.e. Z knows (or A sends it) all the values of $TM(A, B_i)$.

Algorithm 2. Trusted third party dependent

1: A sends around her public key K_A to all $B_i \in S$.
2: Each member B_i computes $C(K_A, DTE(B_i, X))$ and sends it to Z.
3: Z generates a random permutation of all the values received and sends a vector of values to A.
4: A decrypts the encrypted vector of trust in expert values DTE.
5: A sends the vector TM of values $TM(A, B_i)$ to Z.
6: Z permutes the vector with the same permutation used in step (3).
7: A and Z compute the scalar product of DTE and TM vectors using secure computation and obtain the desired sum of products $\rho(A, X)$ (cf. [2] for one possible method to compute scalar product in a secure way).

The main advantage of this method over the previous one, is that individual members do not know A's trust in them. On the other hand the third party Z needs to be trusted with these private values. Moreover, the communication overhead of this scheme is higher, because of the use of secure scalar product, which is an expensive operation (using [2] the complexity is linear but requires the *Add Vectors* protocol which involves many cryptographic transformations).

Scheme 3: Controlled sequence
This scheme, depicted in Algorithm 3, uses the additive scheme described in [9] which enables computing the sum of trust values in an expert, in a private way, independent of any trusted third party. Let us denote the sum of trust values, members in S have in x, as $\tau(S, x)$:

$$\tau(S, x) = \sum_{\forall B_i \in S} DTE(B_i, x)$$

Algorithm 3. Controlled sequence

1: A computes $\tau(S, X)$ using one of the three algorithms described in [9].
2: A decides on a random permutation of members $B_i \in S$ and informs each B_i of the agent following her (next in line) in the permutation.
3: A sends to B_i: $TM'(A, B_i) = (TM(A, B_i) + Q)$, where Q is a random number. We denote $ITE'(A, B_i, x) = TM'(A, B_i) \cdot DTE(Bi, x)$, for simplicity.
4: Using its partial knowledge of A's permutation, B_1 sends the value $ITE'(A, B_1, x)$ to B_2. B_2 sends $ITE'(A, B_1, x) + ITE'(A, B_2, x)$ to B_3. B_3 repeats this process and so do, all the following agents.
The last member sends A the sum $\sum_{\forall B_i \in S} ITE'(A, B_i, x)$.
5: A subtracts $Q \cdot \tau(S, x)$ from the sum it received from the last member, and obtains $\rho(A, x)$ - the desired value.

$$\begin{aligned}\rho(A, x) &= \sum_{\forall B_i \in S} ITE(A, B_i, x) = \sum_{\forall B_i \in S} TM(A, B_i) \cdot DTE(Bi, x) \\ &= \sum_{\forall B_i \in S} (TM(A, B_i) + Q) \cdot DTE(Bi, x) - Q \cdot \sum_{\forall B_i \in S} DTE(B_i, x) \\ &= \sum_{\forall B_i \in S} ITE'(A, B_i, x) - Q \cdot \tau(S, x)\end{aligned}$$

This scheme has the advantage that no trusted third party is involved in it, and it therefore provides more privacy in that respect. However, it has several disadvantages. First, since A is selecting the members it sends the TM' values to, it may select a very small group (e.g. two) thus reducing privacy considerably. This can be overcome by assuring a minimal size to the set of trusted members as discussed in [9] (i.e the witness set). Another way of achieving at least size n group is by having the members use a secret-sharing scheme of at least size n [13], where one of the members must be able to solve the secret. A second disadvantage is that the trust of A in other members, which is a private value, may be compromised by collusion. Any two colluding members $B_i, B_j, i, j \in S$ knowing an approximate value of A's trust in them may be able to approximate Q and find the difference between their respective $TM(A, B_i), TM(A, B_j)$ by a simple subtraction. A way to solve it is to divide the trust set S into two (or more) subsets, which only A knows their composition and use different random numbers for each such subset, again making sure the subset is large enough. Another idea is the following: Let A run the protocol twice. Once with a single Q value, and in the second run, with many different random Qs. Since the users don't know which time is the correct one, their guessing probability is reduced to 50% (and can be reduced further by re-running the protocol for any arbitrary number of times). This can reduce considerably the risk of disclosing

$TM(A, B_j)$ values. The third disadvantage is that the communication overhead in this scheme is even higher than in the second scheme since the complexity of computing $\tau(S, x)$ is $O(n^3)$(see [9]).

3.1 Analysis of Dishonest Users

In the three schemes presented, we consider two forms of dishonesty: the first includes attempts to compromise private data, and the second includes attempts to bias trust values (malicious behavior). A's motivation to act dishonestly is to learn private information (e.g. a member's trust in an expert). Members of A's trust set act dishonestly when they try to find out A's trust in them or if they attempt to bias an expert reputation by providing wrong input.

Members in S gain from acting dishonestly only when A's identity is known to them. We assume that when this information is kept private the agents will follow the above protocols (although members may be strongly biased toward / against expert x, this bias is accepted by A and affects A's trust value in B_i).

In the first scheme members know exactly who the requester A is. If they want to mislead A, they can promote or demote the value $ITE(A, B_i, x)$ they provide to the trusted third party, Z. Also, since each member participating in scheme 1 knows the trust value of A in them, no misconduct is expected in an attempt to infer $TM(A, B)$. However, in this scheme, members may also attempt to gather information about the nature of the relation other members have with A. This is not possible if we assume secure communication, as A sends its information directly to each member, and these only sends their information to Z. In the second scheme, A's identity can be protected. Thus, members have no motivation to mislead A since they don't know who A is. In the third scheme only the first and last users know who A is and A can select these as trusted friends, the rest have no motivation to be malicious.

The requester A may act dishonestly in order to learn a member's direct trust in x by setting the member trust values accordingly (even if we assume that TM of zero is not a valid value). For example, if the trust in a member is a value in the range 1..10 and the trust in an expert is a value in the range 1..5. A may guess the exact trust in an expert by setting one TM value to 1 and the rest to 6. In the first scheme A cannot fake her trust in members of her trust set since we assume that they are aware of A's trust in them. In the second scheme we can use Z to examine the sets of different $TM(A, B_i)$ values that are either suspicious or those that allow A to infer the trust a member has in x with good probability. In the third scheme we can reduce the risk of A faking her trust in members by a verification procedure carried out by members of the trust set that detect very low values as suspicious.

4 Conclusions

In this paper we discussed the problem of computing reputation of members in a community while preserving the privacy of sensitive information. Such information includes the rating of individual members and the trust that one member

has in another. The paper presents three schemes for privacy preserving computation and analyzes their privacy characteristics and communication overhead. The presented schemes apply techniques for secure summation [9] and dot product [2] and use these as primitives in a virtual community oriented setting. Our constructs extends state of the art work to elevate existing trust based models in which privacy is a major concern.

Although the schemes were presented in the context of a specific trust and reputation model, the Knots model, they may be used by other models which take into account the same sensitive information (members ratings and members trust). Future work will include formal proof of the amount of privacy provided, an implemented architecture of the suggested protocols and further discussion of privacy issues in trust and reputation systems.

References

1. eBay, `http://www.ebay.com/`
2. Amirbekyan, A., Estivill-Castro, V.: A new efficient privacy-preserving scalar product protocol (2007)
3. Chakraborty, S., Ray, I.: Trustbac: integrating trust relationships into the rbac model for access control in open systems. In: Proceedings of the eleventh ACM symposium on Access control models and technologies (SACMAT 2006), pp. 49–58. ACM, New York (2006)
4. Gal-Oz, N., Grinshpoun, T., Gudes, E., Meisels, A.: Cross-community reputation: Policies and alternatives (2008)
5. Gal-Oz, N., Gudes, E., Hendler, D.: A robust and knot-aware trust-based reputation model (2008)
6. Hsu, J.Y.-J., Lin, K.-J., Chang, T.-H., Ho, C.-J., Huang, H.-S., Jih, W.-R.: Parameter learning of personalized trust models in broker-based distributed trust management. Information Systems Frontiers 8(4), 321–333 (2006)
7. Josang, A., Ismail, R.: The beta reputation system (2002)
8. Mui, L., Mohtashemi, M., Halberstadt, A.: A computational model of trust and reputation for e-businesses (2002)
9. Pavlov, E., Rosenschein, J.S., Topol, Z.: Supporting privacy in decentralized additive reputation systems (2004)
10. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing (1991)
11. Pingel, F., Steinbrecher, S.: Multilateral secure cross-community reputation systems for internet communities (2008)
12. Resnick, P., Zeckhauser, R.: Trust among strangers in Internet transactions: Empirical analysis of eBay's reputation system. In: The Economics of the Internet and E-Commerce, pp. 127–157 (2002)
13. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
14. Steinbrecher, S.: Design options for privacy-respecting reputation systems within centralised internet communities (2006)
15. Yu, B., Singh, M.: Detecting deception in reputation management (2003)

Building an Application Data Behavior Model for Intrusion Detection

Olivier Sarrouy, Eric Totel, and Bernard Jouga

Supelec, Avenue de la Boulaie, CS 4760, F-35576 Cesson-Sévigné CEDEX, France
firstname.lastname@supelec.fr

Abstract. Application level intrusion detection systems usually rely on the immunological approach. In this approach, the application behavior is compared at runtime with a previously learned application profile of the sequence of system calls it is allowed to emit. Unfortunately, this approach cannot detect anything but control flow violation and thus remains helpless in detecting the attacks that aim pure application data. In this paper, we propose an approach that would enhance the detection of such attacks. Our proposal relies on a data oriented behavioral model that builds the application profile out of dynamically extracted invariant constraints on the application data items.

1 Introduction

Two approaches coexist in intrusion detection. The misuse-based approach relies on the research of known attack signatures in the data collected over the information system. Because it requires signatures, this approach cannot detect unknown attacks. On the other hand, the anomaly-based approach relies on the comparison, at run time, of the application behavior with a previously built normal behavior model. At the application level, the anomaly-based approach is largely preferred. In this case, the normal behavioral model of an application is often built dynamically by the observation of the sequences of system calls it emits [11]. Such system call based intrusion detection systems appear to be insufficient as they cannot detect anything but control flow violations. Thus, attacks that do not disturb the control flow of the application and focus on pure data remain undetected [6].

In this paper we introduce an approach that would enhance the detection of such attacks. Our proposal relies on a data oriented behavioral model. More precisely, our goal is to dynamically discover invariant constraints on the application data that would characterize normal states of the application. This paper is organized as follows : we first discuss the existing work on the topic, we then explain how to build the application data model, and we finally exhibit our prototype implementation before discussing the results we obtain.

2 Related Work

Most recent application level anomaly-based intrusion detection systems rely on the immunological approach introduced by Forrest and al. [11]. This approach is

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 299–306, 2009.

built over an external - or *black-box* - behavioral model and consists in monitoring the sequences of system calls emitted by the application [13]. Such intrusion detection systems are efficient in detecting classical attacks which obviously modify the sequences of system calls emitted by the application, but are easy to evade by mimicing the sequence of system calls the application is supposed to emit [20]. To contend this kind of attacks, proposals have been done to enhance the behavioral model with process internal information, such as the content of the call stack or the value of the program counter at the time of system calls [12]. Such approaches, called *gray-box approaches*, indeed make mimicry attacks more difficult to succeed but remain unable to detect anything but control flow integrity violation. However, another kind of attacks exists which does not disturb the control flow of the application but nonetheless leads to the same threat than classical attacks [18,6].

This second class of attacks focuses on the data that do not influence the application control flow, and are thus called *pure data attacks* or *non-control data attacks*. These attacks constitute an important threat as they cannot be detected by any of the current intrusion detection systems [18]. Furthermore, it appears that numerous real world vulnerabilities can be exploited using such a pure data attack [6]. Some work has already been done which aims at detecting pure data attacks, mainly focusing on the integrity of the application data-flow, either through complete data-flow graph or through taint-checking [4,5,17,14]. In this paper, we propose to dynamically discover likely invariants in the application data in order to characterize its normal behavior. In this goal, we rely on the definition of the set of data that may be sensitive to an intrusion attempt and thus on the notion of tainted data.

3 Pure Data Attacks

As explained in the previous section, current intrusion detection systems remain helpless in detecting what we call pure data attacks, *i.e.*, attacks which do not violate the integrity of the application control flow. In this section we first introduce an example of such an attack and then study it in the perspective of the properties it breaks concerning the data it modifies.

3.1 An Example of a Pure Data Attack

A typical example of a pure data attack may be found in the exploitation of the WU-FTPD *Site Exec Command Format String* Vulnerability [3] described in [6]. This vulnerability allows an attacker to overwrite a `C` structure, denoted `pw`. The severity of this vulnerability resides in that the `pw` structure contains the `uid` of the authenticated user. More precisely, this structure is used to re-affect the application rights to those of the authenticated user after each operation which requires the application to gain root privileges, such as the `setsockopt()` system call used in the treatment of a `GET` request. This example of complete attack, given in Figure 1, is interesting as it clearly does not disturb the control flow of the application, and thus would not be detected by any of the classical system call

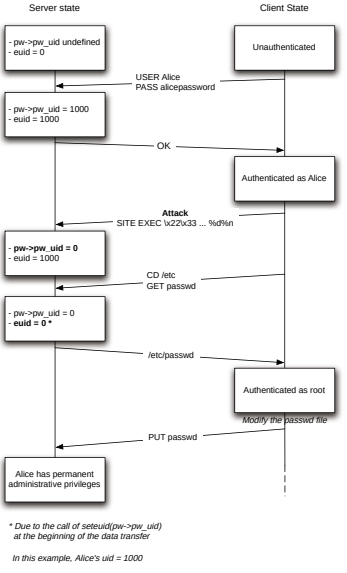


Fig. 1. WU-FTPD pure-data attack

based approaches. An other interesting point in the study of this attack is that it appears to be characteristic of the properties pure data attacks break when they are performed. We discuss this idea in more details in the next subsection.

3.2 Attack Characterization

The pure data attack described in the previous subsection appears to be interesting in how characteristic it is in the disturbance of the expected properties of the application data. Indeed, this attack breaks a very simple property verified during a normal use of the application, which is that the `pw->pw_uid` variable should remain constant during a given session. In this perspective, denoting $\texttt{pw->pw\_uid}_1$, $\texttt{pw->pw\_uid}_2$, ..., $\texttt{pw->pw\_uid}_n$ different values of `pw->pw_uid` during the execution of a given session, this attack may be characterized by the fact that whenever $\texttt{pw->pw\_uid}_1$, $\texttt{pw->pw\_uid}_2$, ..., $\texttt{pw->pw\_uid}_n$ are extracted, it breaks the simple constraint $\texttt{pw->pw\_uid}_1 = \texttt{pw->pw\_uid}_2 = ... = \texttt{pw->pw\_uid}_n$.

More generally, it appears that most pure data attacks - and even more classical attacks - can be characterized by the fact that they break one of the constraints on data that are verified when the application remains in a $\ll$ normal $\gg$ state, *i.e.*, when it runs without being attacked.

4 Data-Based Intrusion Detection Model

As explained above, most pure data attacks - as well as more classical attacks - can be characterized by the fact that they break some properties verified by the

application data when this application remains in a ≪ normal ≫ state. Thus, we believe that extracting and afterwards controlling these properties would allow us to detect intrusions more accurately. In this section, we first introduce a formal definition of a process state and propose an abstraction of this definition focused on the properties verified by the application data. Then, we intend to define the notion of attack in the eyes of this abstraction, still focusing on the properties verified by the application data. Finally, we propose a detection model based on that previously introduced considerations.

4.1 State of a Process

For each discrete time i, the *snapshot state* s_i of a process may be defined by $s_i = \langle pc_i, v_{1_i}, ..., v_{n_i} \rangle$ where pc_i is the location in the process of the executed instruction at time i, i.e. the value of the program counter, and v_{1_i}, ..., v_{n_i} the values of the various data manipulated by the program at time i (registers, environment variables, global and local variables, etc ...). This definition of the state of a process may be insufficient in the perspective of intrusion detection. Indeed, controlling the consistency of snapshot state s_i at a given time i sometimes requires the knowledge of all the previous snapshot states $s_0, ..., s_{i-1}$. We thus define the *global state* S_i of a process - in its temporal meaning - at time i by $S_i = \langle s_0, ..., s_i \rangle = \langle pc_0, ..., pc_i, v_{1_0}, ..., v_{n_0}, ..., v_{1_i}, ..., v_{n_i} \rangle$.

The set of all potential global states of a process, denoted $\mathcal{S}$ is a huge set where all elements cannot be reached during a ≪ normal ≫ execution of a given program. Though, we may define the set of allowed global state of a process, denoted $\mathcal{A}$ as the set of global states which can be reached in a context of an attack free execution.

In practice, it appears that $\mathcal{A}$ constitutes a quite small subset of $\mathcal{S}$ that we would like to define in order to discern the ≪ allowed ≫ states and the ≪ unallowed ≫ states. As it seems impossible and probably not much relevant to explicitly define $\mathcal{A}$, we propose to abstract the state definition to implicitly define it by expressing the various constraints on the values that application data can effectively take when this application is in an allowed global state. In other words, we define a number of relationships which constitute a constraint system $\mathcal{C}$. Given a global state $S_i = \langle s_0, ..., s_i \rangle = \langle pc_0, ..., pc_i, v_{1_0}, ..., v_{n_0}, ..., v_{1_i}, ..., v_{n_i} \rangle$ we consider that $S_i \in \mathcal{A} \iff \langle pc_0, ..., pc_i, v_{1_0}, ..., v_{n_0}, ..., v_{1_i}, ..., v_{n_i} \rangle$ verifies $\mathcal{C}$. Given this definition a ≪ normal ≫ application state, we can now propose an attack model.

4.2 Attack Model

As explained above, we may assume that the application is in an unallowed global state, *i.e.*, a state $S \in \mathcal{S} \backslash \mathcal{A}$ when one of the constraints of $\mathcal{C}$ is broken. Thus, we propose, on the basis of our data constraint based state model, a definition of an attack as a sequence of ≪ user actions ≫ which leads the application from a state $S_i \in \mathcal{A}$ into a state $S_f \in \mathcal{S} \backslash \mathcal{A}$, therefore breaking the constraints on the application data that characterize the set $\mathcal{A}$. Given this definition of an attack, we can now propose a detection model which focuses on the constraints verified by the application data.

4.3 Detection Model

Our detection model relies on the assumption that it is possible to extract in whatever manner the set of constraints $\mathcal{C}$ which characterize the set of allowed global states $\mathcal{A}$. Supposing this assumption verified, we thus propose to monitor the application by controlling the enforcement of this set of constraints $\mathcal{C}$ and to raise an alert as soon as one of these constraints seems broken. However, the extraction of the set of constraints $\mathcal{C}$ may appear quite complex. Nevertheless, as our goal is not to fully qualify the consistency of a given state, but only to qualify it in a security perspective, it is clear that not all the data items manipulated by the application are interesting for our work. In the next section, we thus try to define which data can be critical in a security focused perspective and then examine a way to practically extract this data out of the whole application data set.

5 Intrusion Sensitive Data Set

The set of interesting data we try to extract, henceforward called *intrusion sensitive data set* and noted *ISDS*, is defined by the two main properties it verifies. First, in an immunological approach, it constitutes a subset of the data which may influence the system calls. Furthermore, it constitutes a subset of the data being influenced by user inputs, which we call *tainted data*. The notion of influence between two or more data can be formally defined by the notion of causal dependency [8,19,7]. Denoting *(o,t)* the content of the data object o (a byte or a variable depending on the level of granularity) at time t, we may then denote *(o',t')* $\rightarrow$ *(o,t)*, with $t' \leq t$ the causal dependency of *(o,t)* in relation to *(o',t')*. The relation $\rightarrow$ being transitive, we may then define the causality cone of a point *(o,t)* as $cause(o,t) = \{(o',t')/(o',t') \rightarrow (o,t)\}$. In the same way, we may define the dependency cone *dep(o,t)* as the set of points which causally depend on *(o,t)* and write $dep(o,t) = \{(o',t')/(o,t) \rightarrow (o',t')\}$.

This notion being introduced we may now give a more formal definition of the intrusion sensitive data set. The first property characterizing the intrusion sensitive data set expresses the fact that *ISDS* belongs to the causality cone of the system calls and their arguments. The second property characterizing the intrusion sensitive data set as well expresses the fact that *ISDS* belongs to the dependency cone of user inputs. Thus, as we have expressed *ISDS* as the set of data respecting these two properties, we may define it as the intersection of both the causality cone of the system calls or their arguments and the dependency cone of the user inputs and denote (with *sc* the set of system calls and *ui* the set of user inputs) :

$$ISDS = cause(sc) \cap dep(ui)$$

In the next section, we present the kind of constraints we aim at extracting out of the intrusion sensitive data set we have just defined.

6 Constraints Determination

In section 4.2, we have formulated the hypothesis that attacks generally violate invariant constraints on the application data. We have thus tried to extract these constraints, with the help of an automatic invariant discovery tool called *Daikon* [1,10,9]. Daikon analyzes the execution traces of a given application and tries to extract invariant properties out of it on the basis of a property grammar which contains the set of all searched invariants. These invariants, which are thus exhaustively verified on the execution trace data, can be very complex : constant data item (e.g. x = a), data item taking only a few distinct different values (e.g. $x \in \{a, b, c\}$), definition set (e.g. $x \in [a..b]$), non-nullity, linear relationship (e.g. $x = ay + bz + c$), order relationship (e.g. $x \geq y$), single invariants on x + y, x - y, etc. We introduce in the next section the implementation of our prototype, and explain how to generate the execution traces needed for a Daikon analysis.

7 Implementation

In order to generate the execution traces on *ISDS*, we have used Valgrind [2,16] to emulate a processor controlled by our prototype and have executed the monitored application on this emulated processor. Valgrind is a dynamic binary instrumentation framework offering a dedicated API to emulate a processor and study the binaries executed on this emulated processor. Our intrusion detection system prototype, called *Fatgrind*, has thus been designed as a plugin to Valgrind. Fatgrind mainly aims at dynamically extracting *ISDS* and generating its execution traces containing the value of the data belonging to it. To achieve these goals, Fatgrind builds a shadow of the memory of the monitored process [15]. Each time a byte of memory is written, Valgrind checks whether it is tainted and thus whether it must be shadowed or not. Furthermore, when a system call is emitted, Fatgrind computes its causality cone. Each tainted byte belonging to this causality cone belongs to *ISDS* and is thus dumped into a file. Once enough executions of a given application have been done, the execution traces generated are considered complete enough and are analyzed by *Daikon* to automatically extract the likely invariants it contains. Of course, the quality of the extracted invariants depends on the exhaustivity of the normal application behavior learning.

8 Results

To evaluate our model, we have studied a little snippet of code equivalent to the one studied in section 3. This snippet of code was designed to be representative enough although generating very few traces and thus very few invariant properties, allowing us to easily check their consistency. We have thus focused here on the validation of our approach by trying to control one by one the invariants inferred by Daikon out of the execution traces generated by Fatgrind. Among the various properties extracted we indeed discovered the constraints expressing the

equality of the `uid` variable at the beginning of the daemon loop and at its end. The obtained results have thus shown the viability of the approach on a small but representative example. Moreover it appears that most of the attacks described in [6] would be detected by our approach excepted the attack against the SSH server where the attacked data item is whatever modified during a normal use of the application. This report thus encourages us to pursue the prospective work we have engaged even if a lot of enhancements could be brought to our approach.

9 Conclusion and Future Work

The work presented in this paper proposes a way to enhance application level intrusion detection by introducing a data-oriented detection model. This approach relies on the automatic generation of a behavioral model based on the relationship between application data aiming at detecting state inconsistency at runtime. In order to pursue such an approach, it is necessary to determine which data are sensitive to intrusions and how these data items are related to each other. As shown by the results of the previous section, the proposed approach indeed enables the detection of pure data attacks. However, regarding the conclusion of this prospective work, it appears that several enhancements could be brought. Indeed, the binary level does not allow us to access a high semantical level, as, for instance, we do not get any information about the type of the manipulated data. We therefore plan to apply this approach to programming language using a native intermediate representation (Java, .Net, PHP, etc.) in order to directly modify the interpreter and thus access a richer semantic.

Acknowledgement

This work has been funded by the french DGA (General Delegation for Armament) and the french CNRS (National Center for Scientific Research) in the context of the DALI (Design and Assessment of application Level Intrusion detection system) project.

References

1. Daikon, `groups.csail.mit.edu/pag/daikon/`
2. Valgrind, `http://www.valgrind.org`
3. Cert advisory ca-2001-33 multiple vulnerabilities in wu-ftpd (2001), `http://www.cert.org/advisories/CA-2001-33.html`
4. Castro, M., Costa, M., Harris, T. : Securing software by enforcing data-flow integrity. In : Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (2006)
5. Cavallaro, L., Sekar, R. : Anomalous taint detection. Technical report, Secure Systems Laboratory, Stony Brook University (2008)

6. Chen, S., Xu, J., Sezer, E., Gauriar, P., Iyer, R. : Non-control-data attacks are realistic threats. In : Usenix Security Symposium (2005)
7. d'Ausbourg, B. : Implementing secure dependencies over a network by designing a distributed security subsystem. In : Gollmann, D. (ed.) ESORICS 1994. LNCS, vol. 875. Springer, Heidelberg (1994)
8. Denning, D.E. : A lattice model of secure information flow. Commun. ACM (1976)
9. Ernst, M.D., Cockrell, J., Griswold, W.G., Notkin, D. : Dynamically discovering likely program invariants to support program evolution. IEEE Transactions on Software Engineering (2001)
10. Ernst, M.D., Perkins, J.H., Guo, P.J., McCamant, S., Pacheco, C., Tschantz, M.S., Xiao, C. : The daikon system for dynamic detection of likely invariants. Science of Computer Programming (2007)
11. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A. : A Sense of Self for Unix Processes. In : Proceedings of the 1996 IEEE Symposium on Research in Security and Privacy (1996)
12. Gao, D., Reiter, M.K., Song, D. : Gray-box extraction of execution graphs for anomaly detection. In : Proceedings of the 11th ACM conference on Computer and communications security (2004)
13. Hofmeyr, S.A., Forrest, S., Somayaji, A. : Intrusion detection using sequences of system calls. Journal of Computer Security (1998)
14. Larson, E., Austin, T. : High coverage detection of input-related security faults. In : Proceedings of the 2003 Usenix Conference (Usenix 2003) (2003)
15. Nethercote, N., Seward, J. : How to shadow every byte of memory used by a program. In : Proceedings of the Third International ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments (2007)
16. Nethercote, N., Seward, J. : Valgrind : A framework for heavyweight dynamic binary instrumentation. In : Proceedings of ACM SIGPLAN 2007 Conference on Programming Language Design and Implementation (2007)
17. Newsome, J., Song, D. : Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In : Proceedings of the 12th Annual Network and Distributed System Security Symposium (NDSS 2005) (2005)
18. Parampalli, C., Sekar, R., Johnson, R. : A practical mimicry attack against powerful system-call monitors. Technical report, Secure Systems Laboratory, Stony Brook University (2007)
19. Sabelfeld, A., Myers, A. : Language-based information-flow security (2003)
20. Wagner, D., Soto, P. : Mimicry attacks on host-based intrusion detection systems. In : CCS 2002 : Proceedings of the 9th ACM conference on Computer and communications security (2002)

A Trust-Based Access Control Model for Pervasive Computing Applications

Manachai Toahchoodee, Ramadan Abdunabi, Indrakshi Ray, and Indrajit Ray*

Department of Computer Science,
Colorado State University,
Fort Collins CO 80523-1873

Abstract. With the rapid growth in wireless networks and sensor and mobile devices, we are moving towards an era of pervasive computing. Access control is challenging in these environments. In this work, we propose a trust based approach for access control for pervasive computing systems. Our previously proposed belief based trust model is used to evaluate the trustworthiness of users. Fine-grained access control is achieved depending on the trust levels of users. We develop a class of trust-based access control models having very formal semantics, expressed in graph theory. The models differ with respect to the features they provide, and the types of the trust constraints that they can support.

1 Introduction

Traditional access control models like Mandatory Access Control (MAC), Discretionary Access Control (DAC), and Role-Based Access Control (RBAC) do not work well in pervasive computing systems. Pervasive computing systems are complex, involving rich interactions among various entities. The entities that a system will interact with or the resources that will be accessed are not always known in advance. Thus, it is almost impossible to build a well-defined security perimeter within which the system will operate. Since almost all traditional access control models rely on successful authentication of predefined users, they become unsuitable for pervasive computing systems. Moreover, these systems use the knowledge of surrounding physical spaces to provide services. This requires security policies to use contextual information. For instance, access to a resource may be contingent upon environmental contexts, such as the location of the user and time of day and can be exploited to breach security. Contextual information must, therefore, be protected by appropriate policies.

Researchers have recently started extending the RBAC model to accomodate contextual information such as time and location [1,2,4,5,6,7,10,9,11,12]. However, none of these models address the problem of unknown user in access control. Other researchers have proposed ways to incorporate the concept of trust to RBAC to address this particular problem [13,3]. The general idea in these works is that the access privileges of a user depends on his trust level. However, the applicability of these models to pervasive computing environments remains to be investigated.

* This work was supported in part by the U.S. AFOSR under contract FA9550-07-1-0042.

E. Gudes, J. Vaidya (Eds.): Data and Applications Security 2009, LNCS 5645, pp. 307–314, 2009.

In this paper, we propose a trust-based RBAC model for pervasive computing systems. We adapt the context-sensitive trust model proposed by us earlier [8] in this work. We develop three versions of the model that cater to different circumstances within a system. Users (humans or their representatives and devices) are evaluated for their trustworthiness before they are assigned to different roles. Roles are associated with a trust range indicating the minimum trust level that a user needs to attain before it can be assigned to that role. A permission is also associated with a trust range indicating the minium trust level a user in a specific role needs to attain to activate the permission. The semantics of our model is expressed in graph-theoretic notations that allows us to formulate precise semantics for the model.

The rest of the paper is organized as follows. Section 2 describes how we can evaluate the trust value of the user entity in our model. Section 3 specifies our model using a graphical representation and also presents the different types separation of duty constraints that we can have in our model. Section 4 concludes the paper.

2 Trust Computation

We adapt the trust model proposed by Ray et al. [8]. Initially, an entity A does not trust a new entity B completely. Entity A needs to evaluate a trust relationship with entity B in some context. The context in our model is the role to which a user will be assigned to. We will refer to the context as a role context *rc*. Users can be associated with multiple roles. To determine the authorization between a user and a role, a user's trust value is evaluated based on each role context separately. The trust relationship between human user or device user, and the system in the role context *rc* depends on three factors: *properties*, *experience*, and *recommendations*. The semantics of these three factors are different for the human and the device user. We formally represent a trust relationship between truster, A, and trustee, B, on some role context *rc*, as a triple $({}_Ab_B^{rc}, {}_Ad_B^{rc}, {}_Au_B^{rc})$, where ${}_Ab_B^{rc}$ is A's belief on B about the latter's trustworthiness, ${}_Ad_B^{rc}$ is A's disbelief on B, and ${}_Au_B^{rc}$ is A's uncertainty on B. Each of these components has a value between [0,1] and sum of these components is 1.

A trustee discloses a set of physical properties to be verified by the truster. Examples of such properties for a device are CPU processing speed, memory capacity, transmission rate, signal strength, location of sensor, and physical security. Examples of properties associated with a human user are age, gender, education level, specialization, credentials, and so on. Experience is based on the set of events that had occurred in the past within a certain period of time in which the trustee was involved and that the truster has recollection about. For a device, this can be incidents like number of defects encountered, tamper occurrences, collected data quality, and alarms and control signals responsiveness. For a human user, this could be decisions made in the past, task execution time taken, finesse demonstrated, and so on. Recommendations are provided by trusted third-parties who have knowledge about the trustee with respect to the role context *rc*. Recommendations in case of a device can be provided by other organizations that have used the device under similar circumstances. For a human user, recommendations, for example, can be provided by an organization that he was worked with in the same (or similar) role context *rc*.

Quantifying Properties. Each role in an organization requires certain properties of a user. The properties are scored based on information provided by the user to the system at access request. Each role R is associated with a set of positive properties, $PS_R = \{ps_1, ps_2, \ldots, ps_n\}$, and negative properties $NE_R = \{ne_1, ne_2, \ldots, ne_n\}$, collectively called the role properties. Each positive and negative property is associated with a weight, determined by the organizational policy, that reflects its importance with respect to the role R. Let $w_{ps_1}, w_{ps_2}, \ldots, w_{ps_n}$ be the weights of the positive properties, where $w_{ps_i} \in [0,1]$ and $\sum_{i=1}^{n} w_{ps_i} = 1$. Let $w_{ne_1}, w_{ne_2}, \ldots, w_{ne_n}$ be the weights for negative properties, with $w_{ne_i} \in [0,1]$ and $\sum_{i=1}^{n} w_{ne_i} = 1$.

Let the set of properties possessed by a user B be $UP = up_1, up_2, \ldots, up_n$. Let $p_B = \{UP \bigcap PS_R\}$ be the set of positive properties for the user that are relevant for the role, and $n_B = \{UP \bigcap NE_R\}$ be the set of negative properties. Let w_{ps_i} be the weight of the positive property $p_{B_i} \in UP \bigcap PS_R$, and w_{ne_i} be the weight of the negative property $n_{B_i} \in UP \bigcap NE_R$. Let $m = |UP \bigcap PS_R|$, and $n = |UP \bigcap NE_R|$. The contribution of the user's properties towards its trust is represented by (b_P, d_P, u_P) where b_p, d_p, u_p denotes the belief that the set of properties contribute towards enhancing the opinion about trustworthiness of the trustee, the disbelief that the properties do so, and the uncertainty respectively. Each b_p, d_p and u_p is $\in$ [0, 1] and $b_P + d_P + u_P = 1$. The values of b_P,d_P and u_P are computed using the following formulae:

$$b_P = \frac{\sum_{i=1}^{m} w_{ps_i}}{\sum_{i=1}^{m} w_{ps_i} + \sum_{i=1}^{n} w_{ne_i}}; \quad d_P = \frac{\sum_{i=1}^{n} w_{ne_i}}{\sum_{i=1}^{m} w_{ps_i} + \sum_{i=1}^{n} w_{ne_i}}; \quad \text{and } u_P = 1 - b_P - d_P.$$

Quantifying Experience. We model experience in terms of the number of events encountered by a truster A regarding trustee B in particular context within a specific period of time $[t_0, t_n]$. The time period $[t_0, t_n]$ is equally divided into a set S_i of n intervals, $S_i = \{[t_0, t_1], [t_1, t_2], \ldots, [t_{n-1}, t_n]\}$. The intervals overlap at the boundary points only. The truster A keeps a history file of events performed by the trustee B within these intervals. Within each interval $[t_j, t_{j+1}] \in S_i$ where $j \in \mathbb{N}$, there exists a (possibly empty) set of events that transpired between the user and the system. Events that occurred in the distant past are given less weights than those that occurred more reecntly. We evaluate the experience component of trust, given by the triple (b_E, d_E, u_E), where b_E, d_E and u_E have the same connotation as for properties, in the same manner as in [8].

Quantifying Recommendation. Recommendations play major role on the trust evaluation when the truster does not know much about the trustee. Truster obtains recommendations from one or more recommender that claim to know about the trustee with respect to the particular roles. The recommendation is evaluated based on the recommendations returned by recommender M about B as well as the trust relationship between truster A and the recommender M in providing a recommendation about trustee B. Again we use the same procedure as in [8] to evaluate the recommendation score for the trustee based on a set of recommendations. The recommendation score is given by the triple (b_R, d_R, u_R) with each component having the same connotation as in the evaluation of properties.

Computing Trustworthiness. Using the same ideas as in [8] the trust evaluation policy of the truster is represented by the triple ${}_AW_B^{rc} = (W_P, W_E, W_R)$ where $W_P + W_E + W_R = 1$ and $W_P, W_E, W_R \in [0,1]$. The trust relationship between a truster A and trustee B for a particular role context rc is then given by cross product:

$$(A \xrightarrow{rc} B) = ({}_A b_B^{rc}, {}_A d_B^{rc}, {}_A u_B^{rc}) = (W_P, W_E, W_R) \times \begin{pmatrix} b_P & d_P & u_P \\ b_E & d_E & u_E \\ {}_{AG}b_R & {}_{AG}d_R & {}_{AG}u_R \end{pmatrix}$$

The elements ${}_A b_B^{rc}, {}_A d_B^{rc}, {}_A u_B^{rc} \in [0,1]$, and ${}_A b_B^{rc} +_A d_B^{rc} +_A u_B^{rc} = 1$. After evaluating the trust of the properties, experience, and recommendation factors as earlier the trust value is computed as: $T_{au} = \frac{{}_A b_B^{rc} +_A u_B^{rc}}{{}_A b_B^{rc} +_A d_B^{rc} +_A u_B^{rc}}$. The value T will be in the range of [0,1]. The value closer to 0 indicates low trust value of user B with respect to role R, while the value closer to 1 indicates very high trust value of user with respect to role R.

3 The Trust-Based RBAC Model

We adapt the graph-theoretic approach proposed by Chen and Crampton [5] to define the access control model. The set of vertices $V = U \cup R \cup P$ correspond to the following RBAC entities: (i) Users (U), which can be either human (U_h) or intelligent device (U_d); (ii) Roles (R), which can be categorized to human role (R_h) and device role (R_d), and (iii) Permissions (P), which can be categorized to human permission (P_h) and device permission (P_d). The set of edges $E = UA \cup PA \cup RH_a \cup RH_u$ constitutes of the following: (i) User-Role Assignment (UA) = $(U_h \times R_h) \cup (U_d \times R_d)$ (ii) Permission-Role Assignment (PA) = $(R_h \times P_h) \cup (R_d \times P_d)$ (iii) Role Hierarchy (RH) = $((R_h \times R_h) \cup (R_d \times R_d)) \times \{a,u\}$ consisting of (i) the activation hierarchy (RH_a) = $\{(r,r') : (r,r',a) \in RH\}$, and (ii) the permission usage hierarchy (RH_u) = $\{(r,r') : (r,r',u) \in RH\}$ [(i)]

Trust value for each user is calculated based on the role he has performed previously. The information about the roles the user has performed peviously is stored in the User Role History. The values of trust can be changed from time to time based on user activities. Negative activities such as, committing the fraud in the can decrease his trustworthiness. The calculation process is described in section 2. The system administrator assigns trust constraints in the form of a *trust interval* to roles, permissions, and other associations between entities based on different characteristics of each model. Trust interval is an interval $[l,1]$, where l is the lowest trust value that each role, permission or association is active. (Basically the minimum trust level is specified for each.)

Users of the senior role can perform the same set of duties as its junior role; hence a user who is assigned to the senior role to be more trustworthy than the user who is assigned to the junior role only. Based on this observation we assume that the trust value of the senior role always dominates the trust value of its junior roles. Figure 1 shows the components in our model.

We define the notion of activation path, usage path and access path as follows. An *activation path* (or *act-path*) between v_1 and v_n is defined to be a sequence of vertices $v_1, \ldots, v_n$ such that $(v_1, v_2) \in UA$ and $(v_{i-1}, v_i) \in RH_a$ for $i = 3, \ldots, n$. A *usage path* (or *u-path*) between v_1 and v_n is defined to be a sequence of vertices $v_1, \ldots, v_n$ such that $(v_i, v_{i+1}) \in RH_u$ for $i = 1, \ldots, n-2$, and $(v_{n-1}, v_n) \in PA$. An *access path* (or *acs-path*) between v_1 and v_n is defined to be a sequence of vertices $v_1, \ldots, v_n$, such that (v_1, v_i) is an act-path, and (v_i, v_n) is an u-path. We assume the existence of a trust domain $\mathcal{D}$. The value of trust in the domain can be any real number from zero to one.

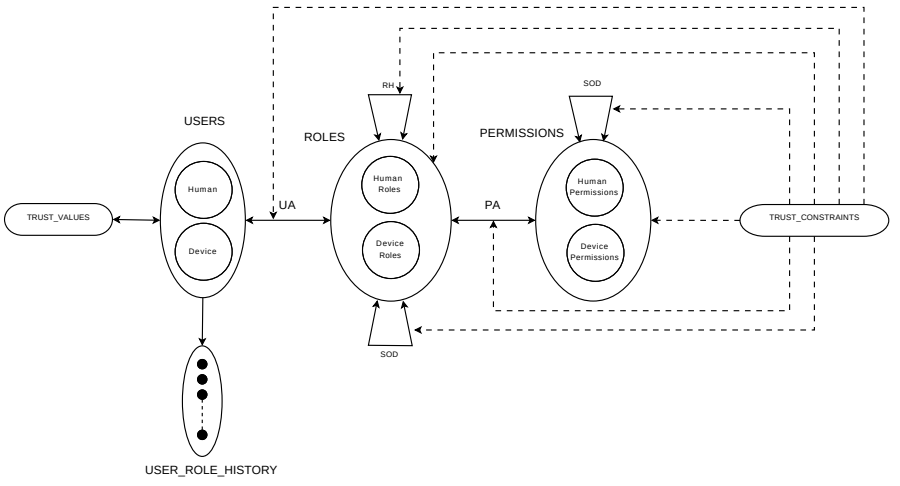


Fig. 1. Trust RBAC Model

The Standard Model. Individual entities, namely, users, roles, and permissions are associated with trust values in the *standard model*. The trust values associated with the user describe how much the user is trusted to perform each specific role. The trust interval associated with a role specify the range of trust values with respect to that role which user has to acquire in order to activate the role. The trust interval associated with a permission specify the minimum trust value with respect to the current role of the user that he has to acquire in order to invoke the permission. The standard model requires that if a user u can invoke a permission p, then the trust value of u is in the trust interval associated with all other nodes in the path connecting u to p. The trust values for the user with respect to each role are denoted with a function $\mathcal{T} : ((U_h \times R_h) \cup (U_d \times R_d)) \rightarrow t \in \mathcal{D}$. The trust interval for role and permission are denoted with a function $\mathcal{L} : (R \cup P) \rightarrow [l,1] \subseteq \mathcal{D}$. For user $u \in U, r \in R$, $\mathcal{T}(u,r)$ denotes the trust value of u with respect to r. For role $r \in R$, $\mathcal{L}(r)$ denotes the trust interval in which r is active. For $p \in P$, $\mathcal{L}(p)$ denotes the trust interval in which p is active. Given a path $v_1,\ldots,v_n$ in the labeled graph $G = (V,E,\mathcal{T},\mathcal{L})$, where $E = UA \cup PA \cup RH_a \cup RH_u$, we write $\hat{\mathcal{L}}(v_2,\ldots,v_n) = \hat{\mathcal{L}}(v_2,v_n) \subseteq \mathcal{D}$ to denote $\bigcap_{i=2}^{n} \mathcal{L}(v_i)$. In other words, $\hat{\mathcal{L}}(v_2,v_n)$ is the trust interval in which every vertex $v_i \in R \cup P$ is enabled.

Authorization in the Standard Model is specified by the following rules: (i) A user $v_1 \in U$ may activate role $v_n \in R$ if and only if there exists an act-path $v_1,v_2,\ldots,v_n$ and $\mathcal{T}(v_1,v_2) \in \mathcal{L}(v_2)$; (ii) A role $v_1 \in R$ is authorized for permission $v_n \in P$ if and only if there exists an u-path $v_1,v_2,\ldots,v_n$ and $\mathcal{L}(v_1) \subseteq \hat{\mathcal{L}}(v_1,v_n)$; (iii) A user $v_1 \in U$ is authorized for permission $v_n \in P$ if and only if there exists an acs-path $v_1,v_2,\ldots,v_i,\ldots,v_n$ such that $v_i \in R$ for some i, $v_1,\ldots,v_i$ is an act-path, $v_i,\ldots,v_n$ is a u-path, v can activate v_i, and v_i is authorized for v'.

The Strong Model. The *strong model* is used when not only the individual entities (users, roles, permissions) involved must satisfy the trust constraints, but the differ-

ent relationships must also satisfy such constraints. For instance, consider the relation $(r,p) \in PA$. In this case, we not only have to take into account the trust values at which the role r can be activated and the trust values at which the permission p can be invoked, but we also must consider the trust values when r can invoke p. This requires specifying another function in the strong model. The trust constraints are denoted with a function $\mu : E \rightarrow [l,1] \subseteq \mathcal{D}$. For $e = (v,v') \in E$, $\mu(v,v')$ denotes the trust interval in which the association between v and v' is active. If $(u,r) \in UA$, then $\mu(u,r)$ denotes the trust interval in which u is assigned to r. If $(r',r) \in RH_a$, then $\mu(r',r)$ denotes the trust interval in which r' is senior to r in the activation hierarchy. If $(r',r) \in RH_u$, then $\mu(r',r)$ denotes the trust interval in which r' is senior to r in the permission usage hierarchy. If $(r,p) \in PA$, then $\mu(r,p)$ denotes the trust interval in which p is assigned to r. Given a path $v_1,\ldots,v_n$ in the labeled graph $G = (V,E,\mathcal{T},\mathcal{L},\mu)$, where $V = U \cup R \cup P$ and $E = UA \cup PA \cup RH_a \cup RH_u$, we write $\hat{\mu}(v_1,\ldots,v_n) = \hat{\mu}(v_1,v_n) \subseteq \mathcal{D}$ to denote $\bigcap_{i=1}^{n-1} \mu(v_i,v_{i+1})$. Hence, $\hat{\mu}(v_1,v_n)$ is the trust interval in which every edge in the path is enabled.

Authorization in the Strong Model is specified by the following rules: (i) a user $v_1 \in U$ may activate role $v_n \in R$ if and only if there exists an act-path $v_1,v_2,\ldots,v_n$ and $\forall i = 2,\ldots,n \bullet \mathcal{T}(v_1,v_i) \in (\mathcal{L}(v_1) \cap \mathcal{L}(v_i) \cap \hat{\mu}(v_1,v_i))$; (ii) a role $v_1 \in R$ is authorized for permission $v_n \in P$ if and only if there exists an u-path $v_1,v_2,\ldots,v_n$ and $\mathcal{L}(v_1) \subseteq (\hat{\mathcal{L}}(v_1,v_n) \cap \hat{\mu}(v_1,v_n))$; (iii) A user $v_1 \in U$ is authorized for permission $v_n \in P$ if and only if there exists an acs-path $v_1,v_2,\ldots,v_i,\ldots,v_n$ such that $v_i \in R$ for some i, $v_1,\ldots,v_i$ is an act-path, $v_i,\ldots,v_n$ is a u-path, v_1 can activate v_i, and v_i is authorized for v_n.

The Weak Model. The *weak model* is derived from the standard model. Recall that the standard model requires that each entity (users, roles, and permissions) in the authorization path be associated with a trust value and in order to be authorized to access other entities, the requester's trust value must be included in the trust interval of the entity he wants to access, together with other entities along the path. In the weak model, the entity v is authorized for another entity v' if the trust value of v is included in the trust interval of v'. There is no requirement that the intermediate nodes on the path satisfy the trust constraints. Like the standard model, the model is based on the labeled graph $G = (V,E,\mathcal{T},\mathcal{L})$, where $V = U \cup R \cup P$ and $E = UA \cup PA \cup RH_a \cup RH_u$.

Authorization in the Weak Model is specified by the following rules: (i) A user $v_1 \in U$ may activate role $v_n \in R$ if and only if there exists an act-path $v_1,v_2,\ldots,v_n$ and $\mathcal{T}(v_1,v_n) \in \mathcal{L}(v_n)$; (ii) A role $v_1 \in R$ is authorized for permission $v_n \in P$ if and only if there exists a u-path $v_1,v_2,\ldots,v_n$ and $\mathcal{L}(v_1) \subseteq \mathcal{L}(v_n)$; (iii) A user $v_1 \in U$ is authorized for permission $v_n \in P$ if and only if there exists an acs-path $v_1,v_2,\ldots,v_i,\ldots,v_n$ such that $v_i \in R$ for some i, $v_1,\ldots,v_i$ is an act-path, $v_i,\ldots,v_n$ is a u-path, v_1 can activate v_i, and v_i is authorized for v_n.

Separation of Duties (SoD) Constraints. Prevent the occurrence of fraud arising out of conflicts of interests in organizations. Separation of duties ensure that conflicting roles are not assigned to the same user or that conflicting permissions are not assigned to the same role.

Separation of Duty (SoD) comes in two varieties, namely, mutual exclusion relations between two roles and between two permissions, denoted by using SD^R and SD^P edges, respectively. The first variety is in order to guarantee that no user can be assigned to two conflicting roles. The second one is to guarantee that no role can be assigned two conflicting permissions. Since SoD is a symmetric relationship, the SD^R and SD^P edges are bi-directional.

The SoDs defined for the standard and weak models are expressed in terms of the graph $G = (V, E, \mathcal{T}, \mathcal{L})$, where $E = UA \cup PA \cup RH_a \cup RH_u \cup SD^R \cup SD^P$ and $V = U \cup R \cup P$. For these cases, the SoD is similar to the SoD constraints in traditional RBAC. These are given below.

SoD Constraints for the Weak and Standard Model

User-Role Assignment: if $(r, r') \in SD^R$ then there are no two edges (u, r) and (u, r') such that $\{(u, r), (u, r')\} \subset UA$

Permission-Role Assignment: if $(p, p') \in SD^P$ then there are no two u-paths of the form $r, v_1, v_2, \ldots, p$ and $r, v'_1, v'_2, \ldots, p'$

Sometimes in the organization we want the user who gain very high trust to be able to bypass the SoDs. For this we define the trust constraint for the separation of duties with a function $\delta : E \rightarrow [l, 1] \subseteq \mathcal{D}$. For $e = (v, v') \in SD^R \cup SD^P$, $\delta(v, v')$ denotes the trust interval in which the SoD constraint can be ignored. In particular,

- if $(r, r') \in SD^R$, $\delta(r, r')$ denotes the trust interval in which the role-role separation of duties constraint can be ignored;
- if $(p, p') \in SD^P$, $\delta(p, p')$ denotes the trust interval in which the permission-permission separation of duties constraint can be ignored.

The strong model is defined over the labeled graph $G = (V, E, \mathcal{T}, \mathcal{L}, \mu, \delta)$, where $E = UA \cup PA \cup RH_a \cup RH_u \cup SD^R \cup SD^P$ and $V = U \cup R \cup P$. The strong model allows specification of weaker forms of SoD constraints than those supported by the traditional RBAC. Specifically, it allows one to specify the trust interval in which the SoD constraints can be ignored.

SoD Constraints for the Strong Model

User-Role Assignment: if $(r, r') \in SD^R$ then there are no two edges (u, r) and (u, r'), corresponding to some user u, where $\mathcal{T}(u, r) \notin (\mathcal{L}(u) \cap \mathcal{L}(r) \cap \mu(u, r) \cap \delta(r, r'))$ and $\mathcal{T}(u, r') \notin (\mathcal{L}(u) \cap \mathcal{L}(r') \cap \mu(u, r') \cap \delta(r, r'))$;

Permission-Role Assignment: if $(p, p') \in SD^P$ then there are no two u-paths $r, v_1, v_2, \ldots, p$ and $r, v'_1, v'_2, \ldots, p'$, where $\mathcal{L}(r) \not\subseteq (\hat{\mathcal{L}}(r, p) \cap \hat{\mu}(r, p) \cap \delta(p, p'))$ and $\mathcal{L}(r) \not\subseteq (\hat{\mathcal{L}}(r, p') \cap \hat{\mu}(r, p') \cap \delta(p, p'))$.

4 Conclusion and Future Work

Traditional access control models are mostly not be suitable for pervasive computing applications. Towards this end, we propose a trust based access control model as an extension of RBAC. We use the context-sensitive model of trust proposed earlier as the underlysing trust model. We investigate the dependence of various entities and relations

in RBAC on trust. This dependency necessitates changes in the invariants and the operations of RBAC. The configuration of the new model is formalized using graph-theoretic notation. In future, we plan to incorporate other environmental contexts, such as space and time, to our model. We also plan to investigate conflicts and redundancies among the constraint specification. Such analysis is needed before our model can be used for real world applications.

References

1. Bertino, E., Bonatti, P., Ferrari, E.: TRBAC: A Temporal Role-Based Access Control Model. In: Proceedings of the 5th ACM Workshop on Role-Based Access Control, Berlin, Germany, pp. 21–30 (2000)
2. Bertino, E., Catania, B., Damiani, M.L., Perlasca, P.: GEO-RBAC: A Spatially Aware RBAC. In: Proceedings of the 10th ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden (2005)
3. Chakraborty, S., Ray, I.: TrustBAC: Integrating Trust Relationships into the RBAC Model for Access Control in Open Systems. In: Proceedings of the 11th ACM Symposium on Access Control Models and Technologies, Lake Tahoe, CA (June 2006)
4. Chandran, S.M., Joshi, J.B.D.: *LoT-RBAC*: A Location and Time-Based RBAC Model. In: Ngu, A.H.H., Kitsuregawa, M., Neuhold, E.J., Chung, J.-Y., Sheng, Q.Z. (eds.) WISE 2005. LNCS, vol. 3806, pp. 361–375. Springer, Heidelberg (2005)
5. Chen, L., Crampton, J.: On Spatio-Temporal Constraints and Inheritance in Role-Based Access Control. In: Proceedings of the ACM Symposium on Information, Computer and Communications Security and Communications Security, Tokyo, Japan (March 2008)
6. Joshi, J.B.D., Bertino, E., Latif, U., Ghafoor, A.: A Generalized Temporal Role-Based Access Control Model. IEEE Transactions on Knowledge and Data Engineering 17(1), 4–23 (2005)
7. Ray, I., Kumar, M., Yu, L.: LRBAC: A Location-Aware Role-Based Access Control Model. In: Bagchi, A., Atluri, V. (eds.) ICISS 2006. LNCS, vol. 4332, pp. 147–161. Springer, Heidelberg (2006)
8. Ray, I., Ray, I., Chakraborty, S.: An Interoperable Context Sensitive Model of Trust. Journal of Intelligent Information Systems 32(1), 75–104 (2009)
9. Ray, I., Toahchoodee, M.: A Spatio-Temporal Access Control Model Supporting Delegation for Pervasive Computing Applications. In: Proceedings of the 5th International Conference on Trust, Privacy & Security in Digital Business, Turin, Italy (September 2008)
10. Sampemane, G., Naldurg, P., Campbell, R.H.: Access Control for Active Spaces. In: Proceedings of the Annual Computer Security Applications Conference, Las Vegas, NV, USA (December 2002)
11. Samuel, A., Ghafoor, A., Bertino, E.: A Framework for Specification and Verification of Generalized Spatio-Temporal Role Based Access Control Model. Technical Report CERIAS TR 2007-08, Purdue University (February 2007)
12. Toahchoodee, M., Ray, I.: On the Formal Analysis of a Spatio-Temporal Role-Based Access Control Model. In: Atluri, V. (ed.) DAS 2008. LNCS, vol. 5094, pp. 17–32. Springer, Heidelberg (2008)
13. Ya-Jun, G., Fan, H., Qing-Guo, Z., Rong, L.: An Access Control Model for Ubiquitous Computing Application. In: Proceedings of the 2nd International Conference on Mobile Technology, Applications and Systems, Guangzhou, China (November 2005)

Author Index