LNCS 4602

Steve Barker
Gail-Joon Ahn (Eds.)

Data and Applications Security XXI

21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security
Redondo Beach, CA, USA, July 2007, Proceedings

ifip

Springer

Steve Barker   Gail-Joon Ahn (Eds.)

Data and Applications Security XXI

21st Annual IFIP WG 11.3 Working Conference on
Data and Applications Security
Redondo Beach, CA, USA, July 8-11, 2007
Proceedings

Springer

Volume Editors

Steve Barker
King's College London, Department of Computer Science
Strand, London WC2R 2LS, UK
E-mail: steve.barker@kcl.ac.uk

Gail-Joon Ahn
University of North Carolina at Charlotte
Department of Software and Information Systems
9201 University City Blvd., Charlotte, NC 28223, USA
E-mail: gahn@uncc.edu

Library of Congress Control Number: 2007299928

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 3-540-73533-X Springer Berlin Heidelberg New York
ISBN-13 978-3-540-73533-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12088249 06/3180 5 4 3 2 1 0

Preface

This volume contains the papers presented at the 21st Annual IFIP WG 11.3 Conference on Data and Applications Security (DBSEC) held July 8–11 in Redondo Beach, California, USA. The purpose of the DBSEC conference is to disseminate original research results and experience reports in data and applications security.

In response to the call for papers, 44 submissions were received. Following a rigorous reviewing process, 18 high-quality papers were accepted for presentation and publication. In addition, two short papers were selected for poster presentation. The conference program also included one invited talk and one panel discussion. We believe that the program includes a balanced mix of practical experiences and theoretical results that consolidate existing work and suggest emerging areas of interest for researchers in data and applications security.

The continued success of the DBSEC conference is due to the efforts of many individuals. We would like to thank all of the researchers that submitted papers to DBSEC for consideration, and the Program Committee members and additional reviewers for making the review process fair and thorough, and for providing valuable suggestions on each submission. We are also indebted to the invited speakers and panelists for their contributions to the success of the conference.

We would also like to thank our General Chair, Sharad Mehrotra, and our Local Arrangements Chair, Chen Li. Special thanks go to our Publicity Chair, Paul Douglas, for advertising the conference, to the IFIP WG 11.3 Chair, Pierangela Samarati, and to Sushil Jajodia for their help and support in organizing the conference.

July 2007

Steve Barker
Gail-Joon Ahn

Organization

Executive Committee

General Chair	Sharad Mehrotra (University of California, Irvine, USA)
Program Chairs	Steve Barker (King's College, London University, UK)
	Gail-Joon Ahn (University of North Carolina at Charlotte, USA)
Publicity Chair	Paul Douglas (University of Westminster, UK)
Local Arrangements	Chen Li (University of California, Irvine, USA)
IFIP WG 11.3 Chair	Pierangela Samarati (Universita degli Studi di Milano, Italy)

Program Committee

Anne Anderson	Sun Microsystems, USA
Vijay Atluri	Rutgers University, USA
Sabrina De Capitani di Vimercati	Università degli Studi di Milano, Italy
Soon Ae Chun	City University of New York, USA
Chris Clifton	Purdue University, USA
Jason Crampton	Royal Holloway, London University, UK
Steve Demurjian	University of Connecticut, USA
Csilla Farkas	University of South Carolina, USA
Eduardo Fernandez-Medina	University of Castilla-La Mancha, Spain
Qijun Gu	Texas State University, USA
Ehud Gudes	Ben-Gurion University, Israel
Sushil Jajodia	Gearge Mason University, USA
Carl Landwehr	University of Maryland, USA
Peng Liu	Pennsylvania State University, USA
Patrick McDaniel	Pennsylvania State University, USA
Ravi Mukkamala	Old Dominion University, USA
Peng Ning	North Carolina State University, USA
Sylvia Osborn	University of Western Ontario, Canada
Brajendra Panda	University of Arkansas, USA
Jaehong Park	Eastern Michigan University, USA
Joon Park	Syracuse University, USA
Indrakshi Ray	Colorado State University, USA
Indrajit Ray	Colorado State University, USA
Pierangela Samarati	Università degli Studi di Milano, Italy
Ravi Sandhu	George Mason University, USA
Andreas Schaad	SAP Laboratories, France
Dongwan Shin	New Mexico Tech, USA

Anoop Singhal	NIST, USA
David Spooner	Rennselaer Polytechnic Institute, USA
Bhavani Thuraisingham	University of Texas Dallas, USA
T.C. Ting	University of Connecticut, USA
Duminda Wijesekera	George Mason University, USA
Meng Yu	Monmouth University, USA
Ting Yu	North Carolina State University, USA

Table of Contents

Confidentiality Policies for Controlled Query Evaluation

Joachim Biskup and Torben Weibert*

Fachbereich Informatik, Universität Dortmund, 44221 Dortmund, Germany
{biskup,weibert}@ls6.cs.uni-dortmund.de

Abstract. Controlled Query Evaluation (CQE) is an approach to enforcing confidentiality in information systems at runtime. At each query, a censor checks whether the answer to that query would enable the user to infer any information he is not allowed to know according to some specified confidentiality policy. If this is the case, the answer is distorted, either by refusing to answer or by returning a modified answer. In this paper, we consider incomplete logic databases and investigate the semantic ways of protecting a piece of information. We give a formal definition of such confidentiality policies, and show how to enforce them by reusing the existing methods for CQE.

Keywords: Inference control, confidentiality policies, logic databases.

1 Introduction

Security in information systems aims at various goals, one of which is preservation of *confidentiality*: Certain information may only be disclosed to a certain subgroup of users. This is of particular importance when an information system contains both classified and public data, and is accessed by multiple users at the same time. Confidentiality can be achieved by various methods, which can be divided into two categories: *access control*, which us usually implemented by static access rights, and *information flow control*, which is often applied dynamically at query time. The latter addresses the *inference problem*: A user might combine multiple pieces of (public) information in order to infer secret information. The inference problem has been studied in a various contexts, for example statistical (see [1,2,3] for an introduction and e. g. [4,5,6] for more recent work), multi-level and relational databases (see e. g. [7,1,8,9,10,11]). See [12] for a comprehensive review of the respective approaches.

Controlled Query Evaluation (CQE) is a dynamic approach for information flow control in logic databases, which are either *complete* (i. e., they can provide an answer to each query) or *incomplete* (i. e., part of the information is missing, and some queries cannot be answered). The administrator defines a *confidentiality policy*, specifying the information to be kept secret. At runtime, before an

* This author is funded by the German Research Foundation (DFG) under Grant No. BI-311/12-1.

answer to a query is returned to the user, it is passed to a *censor* which investigates possible security risks. In order to identify these risks, a *log file* of past queries and answers is maintained. In case the answer would reveal any secret information (either directly or combined with previous answers), the answer is *distorted*, either by *lying* (giving a "false" answer) or by *refusal* (returning no "useful" answer at all). CQE was first proposed by Sicherman et at. [13] and Bonatti et al. [14]. A unified framework for *complete* databases was introduced by Biskup [15] and later exploited by Biskup/Bonatti to investigate CQE under various parameters [16,17,18,19]. Some of these parameters have also been investigated for incomplete databases [20,21]. This paper extends the work on incomplete databases, filling some of the gaps.

A database instance *db* is a consistent set of sentences of some logic (in this paper, propositional logic); a closed (yes-no) query Φ is a single sentence of that logic, and its value in a database instance *db* is either *true*, *false* or *undef*. A *potential secret* is a sentence Ψ. In case Ψ is *true* in *db*, the user may not infer this fact; otherwise, if Ψ is either *false* or *undef*, this fact may be disclosed. Thus, a potential secret protects the fact *that* some information is *true*.

For complete information systems, another type of confidentiality policies has been studied: *secrecies*. A secrecy is a pair of complementary sentences $(\Psi, \neg\Psi)$. CQE will conceal *whether* Ψ or $\neg\Psi$ holds in the database; as opposed to potential secrets, the negation is protected as well. As discussed in [18], secrecies can be protected by discretely designed enforcement methods, or by "naively" reducing them into a set of potential secrets $\{\Psi, \neg\Psi\}$, and then reusing the existing enforcement methods for potential secrets.

In this paper, we investigate whether the concept of secrecies can be adopted for incomplete information systems as well. As it turns out, incomplete information systems offer many different semantic ways of protecting a sentence Ψ. For example, it is possible to protect the partial information that "Ψ is either *true* or *false*, but not *undef*"; or one might want to keep the user from inferring *any* information about the actual value of Ψ. We show how these "generalized" confidentiality targets can be formalized, and how they can be operationally enforced by reduction to existing techniques.

In Section 2, we recall CQE for incomplete databases and potential secrets, as found in [21]. We also present an example enforcement method which can later be used for the reduction. Section 3 discusses the various ways of protecting secret information under incomplete databases, and gives a formal definition of generalized confidentiality policies. In Section 4, we demonstrate the reduction to potential secrets, and discuss the requirements for the underlying enforcement method. We finally conclude in Section 5.

2 Controlled Query Evaluation for Potential Secrets

In this section, we summarize the CQE framework for potential secrets from [21]. We first specify the abstract framework and its declarative notion of confidentiality, and then present an instantiation thereof, the *combined lying and refusal method*.

2.1 Declarative Framework

We consider (possibly) incomplete logic databases, based on propositional logic.

Definition 1. *A database schema DS is a finite set of propositions. The propositional language over DS is denoted by $\mathcal{P}_{DS}$. A database instance over the schema DS is a consistent set $db \subset \mathcal{P}_{DS}$ of propositional sentences. A query $\Phi \in \mathcal{P}_{DS}$ is a propositional sentence. The result of a query Φ in a database instance db is determined by the function*

$$eval(\Phi)(db) := \begin{cases} true & \text{if } db \models_{PL} \Phi \\ false & \text{if } db \models_{PL} \neg\Phi \\ undef & \text{otherwise} \end{cases} \tag{1}$$

(where $\models_{PL}$ denotes logical implication in propositional logic). We assume that the user does not issue a single query but a sequence of queries $Q = \langle \Phi_1, \ldots, \Phi_n \rangle$. (The framework might be extended to a fragment of first-order logic, given that logical implication is decidable in that fragment; see [16] for a discussion.)

In previous work, CQE for incomplete databases has been studied for *potential secrets.*

Definition 2. *A confidentiality policy based on potential secrets is a set $pot\_sec = \{\Psi_1, \ldots, \Psi_m\}$ of propositional sentences, each of which we call a potential secret. The semantics of the confidentiality policy is as follows: In case Ψ_i is true in the actual database instance db, the user is not allowed to infer this information. On the other hand, if Ψ_i is either false or undef in db, this information may be disclosed.*

Potential secrets are a suitable formalization for real-life situations where the circumstance *that* a certain fact is true must be kept secret, but not the converse.

Example 3. Imagine a person applying for an employment. If that person suffers from a terminal disease, this fact must be kept secret. On the other hand, if the applicant is healthy, this information may be disclosed. The sentence "person X suffers from a terminal disease" can be formalized as a potential secret Ψ.

The confidentiality policy is declared independently from the actual database instance *db*, and may contain both potential secrets that are true in *db*, and potential secrets that are not true in *db*. This is important as we assume that the user *knows* the set of potential secrets (but of course not their respective values in *db*). CQE enforces the confidentiality policy by iteratively examining each query and the inferences the user could draw from the respective answer. In case confidentiality is threatened, a modified answer is given, in one of two possible ways:

1. *Lying*: An answer different from the actual query value is returned, for example *false* instead of *true*.
2. *Refusal*: Instead of the actual query value, the special answer *refuse* is returned.

CQE also accounts for any information known or assumed by the user prior to the first query, for example general knowledge or publicly known semantic constraints. These *a priori assumptions* are formalized as a set *prior* of propositional sentences.

All things considered, a CQE method for potential secrets can be formalized as a function $cqe(Q, db, prior, pot\_sec) := \langle ans_1, \dots, ans_n \rangle$, where $Q = \langle \Phi_1, ..., \Phi_n \rangle$ is a query sequence, *prior* are the a priori assumptions, *db* is a database instance, and *pot_sec* is a set of potential secrets. The output is a sequence of answers ans_i. Each enforcement method *cqe* goes along with a function *precondition* that defines the admissible arguments $(prior, db, pot\_sec)$ for that method. In particular, *precondition* makes sure that *prior* does not imply any potential secret in the first place. The system will reject to start a session unless *precondition* is satisfied.

Definition 4. *Let cqe be a CQE method for potential secrets, with precondition defining the admissible arguments. cqe is defined to preserve confidentiality iff*

for all finite query sequences Q,
for all confidentiality policies pot_sec,
for all potential secrets $\Psi \in pot\_sec$,
for all a priori assumptions $prior$,
for all instances db_1 *so that* $precondition(db_1, prior, pot\_sec)$ *holds*
there exists an instance db_2
so that $precondition(db_2, prior, pot\_sec)$ *holds and*
(a) [db_1 *and* db_2 *produce the same answers]*
$cqe(Q, db_1, prior, pot\_sec) = cqe(Q, db_2, prior, pot\_sec)$
(b) [Ψ *is not true in* db_2*]*
$eval(\Psi)(db_2) \in \{false, undef\}$.

Condition (b) ensures that there is an instance db_2 in which Ψ is not *true*. Condition (a) guarantees that db_1 and db_2 produce the same answers; the user cannot distinguish db_1 from db_2, and thus cannot rule out that Ψ is actually *false* or *undef*. This confidentiality definition is purely declarative. In the following, we show how to operationally meet these requirements by keeping a log file of sentences in epistemic logic.

2.2 An Enforcement Method with Lying and Refusal

We outline the *combined lying and refusal* approach from [21]. In order to account for the information disclosed by previous answers, the system keeps a *log file* log_i as a set of sentences in *epistemic logic*. This logic, also known as S5 modal logic, is established by introducing the modal operator K which we read as "the database knows that…". The resulting language, based on a set DS of propositions, is denoted by $\mathcal{L}_{DS}$. We use the common Kripke-style semantics, to be found e. g. in [22]: An $\mathcal{M}_{DS}$-structure is a triple $M = (S, \mathcal{K}, \pi)$, where S is a set of states, $\mathcal{K}$ a binary equivalence relation on S, and $\pi : S \times DS \to \{\mathit{true}, \mathit{false}\}$ assigns a truth value to each proposition from DS under each state $s \in S$. The semantics of the K operator is defined by

$$(M, s) \models K\Phi \text{ iff } (M, s') \models \Phi \text{ for all } s' \text{ such that } (s, s') \in \mathcal{K} \tag{2}$$

(where $\models$ is the ordinary model-of operator). A sentence ϕ is logically implied by a set of sentences Σ wrt. $\mathcal{M}_{DS}$ (in formulae: $\Sigma \models_{S5} \phi$) iff for every $\mathcal{M}_{DS}$-structure $M = (S, \mathcal{K}, \pi)$ and every state $s \in S$ it holds that if $(M, s) \models \Sigma$ then $(M, s) \models \phi$.

A propositional sentence ϕ and a truth value $v \in \{\mathit{true}, \mathit{false}, \mathit{undef}\}$ can be converted into an appropriate epistemic sentence by the function Δ with

$$\begin{aligned}
&\Delta(\phi, \mathit{true}) = K\phi, \\
&\Delta(\phi, \mathit{false}) = K\neg\phi, \\
&\Delta(\phi, \mathit{undef}) = \neg K\phi \wedge \neg K\neg\phi.
\end{aligned}$$

Furthermore, we define the function

$$\Delta^*(\phi, V) := \bigvee_{v \in V} \Delta(\phi, v)$$

that converts a sentence ϕ and a non-empty set of values $\emptyset \neq V \subseteq \{\mathit{true}, \mathit{false}, \mathit{undef}\}$ into an epistemic sentence by disjunctively connecting the single sentences $\Delta(\phi, v)$ for each $v \in V$. The set V is also called an *inference set*, as it is used to formalize (disjunctive) information about a query value. For example, $V = \{\mathit{true}, \mathit{undef}\}$ means "the query value is either *true* or *undef*, but not *false*". A unary inference set represents definitive information (exactly one value appears possible), a binary inference set disjunctive information (two values appear possible, one does not), and the inference set $\{\mathit{true}, \mathit{false}, \mathit{undef}\}$ represents no information (any value appears possible). In particular, note that $\Delta^*(\phi, \{\mathit{true}, \mathit{false}, \mathit{undef}\}) = K\phi \;\vee\; K\neg\phi \;\vee\; \neg K\phi \wedge \neg K\neg\phi$ is a tautology.

Prior to the first query, the log file is initialized with the a priori assumptions: $\mathit{log}_0 := \mathit{prior}$. Later, after each query Φ_i, log_i is established by translating the information disclosed by the i-th answer ans_i into an epistemic sentence, and adding it to log_{i-1}. In case of a regular answer $\mathit{ans}_i \in \{\mathit{true}, \mathit{false}, \mathit{undef}\}$ (being a lie or not), the translation $\Delta^*(\Phi_i, \{\mathit{ans}_i\})$ of the definite inference set $\{\mathit{ans}_i\}$ is added to the log file. In case the answer was refused ($\mathit{ans}_i = \mathit{refuse}$), it is assumed to provide no information to the user, so the tautology $\Delta^*(\Phi_i, \{\mathit{true}, \mathit{false}, \mathit{undef}\})$ is added.

Having formalized the previous knowledge as epistemic sentences, we can employ logical implication in order to detect confidentiality violations: A potential secret $\Psi \in \mathit{pot\_sec}$ is considered disclosed if it is logically implied by the log file log_i. The goal is to prevent these violations throughout the query sequence. We formalize the combined lying and refusal approach as a function $cqe_{combined}(Q, db, prior, pot\_sec)$ with the precondition

$$precondition_{combined}(db, prior, pot\_sec) \ := \ (\forall \Psi \in pot\_sec)[\ prior \not\models_{S5} \Psi\ ],$$

which prevents that the a priori assumptions already lead to a violation. After each query Φ_i, the returned answer ans_i and the internal log file log_i are generated as follows:

1. Determine the *security configuration*, i. e., the set of definitive inferences that would lead to the disclosure of at least one potential secret:

$$C_i := \{\ V \in \{\{true\}, \{false\}, \{undef\}\} \mid (\exists \Psi \in pot\_sec)\ [\ log_{i-1} \cup \{\Delta^*(\Phi_i, V)\} \models_{S5} \Psi\ ]\ \} \quad (3)$$

2. Use a *censor function* to choose the answer $ans_i \in \{true, false, undef, refuse\}$ to return, according to the security configuration C_i and the actual query value $eval(\Phi_i)(db)$. The censor function must meet certain requirements; in particular, it must make sure that $\{ans_i\} \notin C_i$. An example of an appropriate censor function is given in Table 1. Black cells indicate a modified answer.

Table 1. Censor function for the combined lying and refusal method

Security Configuration	$eval(\Phi)(db) = ...$		
C	*true*	*false*	*undef*
$\{\{true\}, \{false\}, \{undef\}\}$	*refuse*	*refuse*	*refuse*
$\{\{true\}, \{false\}\}$	*undef*	*undef*	*undef*
$\{\{true\}, \{undef\}\}$	*false*	*false*	*false*
$\{\{false\}, \{undef\}\}$	*true*	*true*	*true*
$\{\{true\}\}$	*undef*	*false*	*undef*
$\{\{false\}\}$	*true*	*undef*	*undef*
$\{\{undef\}\}$	*true*	*false*	*false*
$\emptyset$	*true*	*false*	*undef*

3. Update the log file by adding the answer translated into an epistemic sentence:

$$log_i := \begin{cases} log_{i-1} \cup \{\Delta^*(\Phi, \{true, false, undef\})\} & \text{if } ans_i = refuse \\ log_{i-1} \cup \{\Delta^*(\Phi, \{ans_i\})\} & \text{otherwise} \end{cases}$$

Theorem 5. *$cqe_{combined}$ preserves confidentiality according to Definition 4.*

The full proof can be found in [21], but we give a short sketch here.

First, it can be shown by induction that $log_i \not\models_{S5} \Psi$ holds for all $\Psi \in pot\_sec$ and all $1 \leq i \leq n$, in particular for the final log file log_n. Thus, given a potential secret $\Psi \in \mathit{pot\_sec}$, there must be an $\mathcal{M}_{DS}$-structure $M = (S, \mathcal{K}, \pi)$ and a state $s \in S$ such that

$$(M, s) \models \mathit{log}_n \text{ but } (M, s) \not\models \Psi.$$

An alternative instance db_2 can be constructed from (M, s) by

$$db_2 := \{\ \alpha \mid \alpha \text{ is a propositional sentence and } (M, s) \models K\alpha\ \}. \tag{4}$$

db_2 is consistent, so it is a valid database instance, and it is also closed under logical implication.

As $(M, s) \not\models \Psi$, we conclude that Ψ cannot be *true* in db_2. Finally, it can be shown that the same answers are generated under both db_2 and the original instance db_1.

3 Generalized Confidentiality Policies

Controlled Query Evaluation for incomplete databases, as summarized in Section 2, protects a set of potential secrets; for each potential secret, the user may not infer that this secret is true in the actual database instance. Example 3 demonstrates that potential secrets have a useful semantics in many situations. However, there are situations in which a sentence must be protected in a different semantic way.

Example 6. For the sake of sexual equality, an applicant's gender may not have an influence on whether he or she is chosen for a particular job. Hence, a person querying a database containing applicants' data may not infer that a given person is male, and neither that this person is *not* male.

Example 7. Although being supposed to do housework, Jim secretly goes to watch his favorite team's soccer match. Talking to his wife later, Jim must keep secret whether his team won or or not. Furthermore, his wife must not even learn that Jim knows whether his team won or not, as this would disclose the fact that he went to the match.

A *confidentiality target* consists of two parts: The sentence that is to be protected, and the set of truth values the user is not allowed to infer. The latter part can be definite ("the team has won") or partial ("the team has won or has not won"[1]).

Definition 8. *A* confidentiality target *is a pair (ψ, V), where ψ is a propositional sentence, and $V_i \subset \{\mathit{true}, \mathit{false}, \mathit{undef}\}$ with $\emptyset \neq V_i \neq \{\mathit{true}, \mathit{false}, \mathit{undef}\}$ is a non-empty inference set.*[2] *A* (generalized) confidentiality policy *is a set $\mathit{policy} = \{(\psi_1, V_1), \ldots, (\psi_m, V_m)\}$ of confidentiality targets.*

[1] This is not a tautology due to the remaining alternative "he does not know whether the team won".

[2] It is reasonable to prohibit $\{\mathit{true}, \mathit{false}, \mathit{undef}\}$, as this is a tautology and can never be protected. The empty set does not make sense either, as at least one value needs to be protected.

Example 9. Given two propositional sentences a and b, the confidentiality policy given by $\mathit{policy} = \{\ (a, \{\mathit{true}, \mathit{undef}\}),\ (b, \{\mathit{false}\})\ \}$ declares that (1) the user may not infer that a is either *true* or *undef*, and that (2) the user may not infer that b is *false*.

Example 10. Consider the propositional sentence a and the confidentiality policy given by $\mathit{policy} = \{\ (a, \{\mathit{true}, \mathit{false}\}), (a, \{\mathit{true}, \mathit{undef}\}), (a, \{\mathit{false}, \mathit{undef}\})\ \}$. Obviously, any disjunctive information is considered harmful, so the user may not learn any information about the value of a at all. This corresponds to the concept of *secrecies* investigated in the context of complete databases [15,18], where the user may not learn the exact value of some sentence.

We can formalize a CQE method for generalized confidentiality policies as a function

$$\mathit{cqe}^*(Q, \mathit{db}, \mathit{prior}, \mathit{policy}) := \langle \mathit{ans}_1, \ldots, \mathit{ans}_n \rangle,$$

where $Q = \langle \Phi_1, \ldots, \Phi_n \rangle$ is the query sequence, *prior* is the set of a priori assumptions, *db* is the database instance and *policy* is the confidentiality policy. Each method goes along with a function $\mathit{precondition}^*$ that defines the admissible arguments.

Definition 11. *Let cqe^* be a CQE method for generalized confidentiality policies with $\mathit{precondition}^*$ as its associated precondition for admissible arguments. cqe^* is defined to preserve confidentiality iff*

for all finite query sequences Q,
for all generalized confidentiality policies policy,
for all confidentiality targets $(\psi, V) \in \mathit{policy}$,
for all a priori assumptions prior,
for all instances db_1 so that $\mathit{precondition}^*(\mathit{db}_1, \mathit{prior}, \mathit{pot\_sec})$ holds
there exists an instance db_2
so that $\mathit{precondition}^*(\mathit{db}_2, \mathit{prior}, \mathit{pot\_sec})$ holds and
(a) [db_1 and db_2 produce the same answers]
$\mathit{cqe}^*(Q, \mathit{db}_1, \mathit{prior}, \mathit{policy}) = \mathit{cqe}^*(Q, \mathit{db}_2, \mathit{prior}, \mathit{policy})$
(b) [ψ has a “permitted” value in db_2]
$\mathit{eval}(\psi)(\mathit{db}_2) \notin V$.

Again, this definition is purely declarative. In the following section, we will show how to operationally meet these requirements, reusing existing techniques.

One advantage of the new concept of confidentiality targets is that they have a very simple syntax, which allows easy declaration of confidentiality policies. However, we run into problems when we want to design an operational enforcement method for this kind of confidentiality policies – there is no logical implication operator defined for this language, and of course there are no proof systems available that could be used in an implementation.

In the following section, we will present a solution to this problem: Each confidentiality target can be converted into a single sentence of modal epistemic logic. These sentences can then be regarded as (epistemic) potential secrets, and we can reuse the existing methods for potential secrets in order to enforce the converted confidentiality policy.

4 Enforcement by Reduction

In the previous section, we gave a declarative definition of confidentiality wrt. generalized policies. We will now show how to enforce these policies by reusing the methods established for potential secrets. The idea is to convert the generalized confidentiality policy into a set of potential secrets. As facts like "the value is either *true* or *false*" or "the value is *undef*" need to be protected, it is necessary to use an epistemic representation of the confidentiality targets.

Definition 12. *Let* $policy = \{(\psi_1, V_1), \ldots, (\psi_n, V_m)\}$ *be a (generalized) confidentiality policy. policy can be converted into a set of (epistemic) potential secrets by the function*

$$pot\_sec(policy) := \{\Delta^*(\psi_1, V_1), \ldots, \Delta^*(\psi_m, V_m)\} \tag{5}$$

where Δ^* *is the conversion function defined in Section 2.2.*

Remember that, according to Definition 8, the value set V_i of each confidentiality target $(\psi_i, V_i) \in policy$ is either unary or binary. Thus, all sentences in $pot\_sec(policy)$ have one of the following six syntactic forms, where ψ is a propositional sentence:

(1) $K\psi$,
(2) $K\neg\psi$,
(3) $\neg K\psi \wedge \neg K\neg\psi$,
(4) $K\psi \;\vee\; K\neg\psi$,
(5) $K\psi \;\vee\; \neg K\psi \wedge \neg K\neg\psi$,
(6) $K\neg\psi \;\vee\; \neg K\psi \wedge \neg K\neg\psi$.

Example 13. The confidentiality policy given by

$$policy \;=\; \{\; (a, \{true, undef\}),\; (b, \{false\})\; \}$$

is converted into

$$pot\_sec(policy) = \;\{\; Ka \;\;\vee\;\; \neg Ka \wedge \neg K\neg a, \quad K\neg b\; \}.$$

The remaining problem is that the CQE methods for potential secrets, as defined in Section 2, only allow *pot_sec* to contain propositional sentences. However, as the epistemic language is a superset of the propositional language, some enforcement methods might also work for epistemic potential secrets. Given a "useful" behavior, these methods would then be exploitable for the conversion of confidentiality targets. We will first give a formal definition of these two requirements – suitable for epistemic potential secrets, and "useful" behavior – and then prove that $cqe_{combined}$ satisfies these properties.

Definition 14. *An enforcement method cqe wrt. potential secrets is* adapted for epistemic potential secrets *iff*

1. *the specific algorithm of cqe accepts a set of* epistemic sentences *(instead of propositional sentences) to be passed as the pot_sec input parameter, and*
2. *given a propositional sentence ψ and an epistemic potential secret Ψ associated with ψ, there exists an alternative database instance db_2 as demanded by Definition 4 that has the following properties wrt. the value of ψ:*

epistemic potential secret Ψ	*value of ψ in db_2*
$\Delta^*(\psi, \{true\}) = K\psi$	$eval(\psi)(db_2) \in \{false, undef\}$
$\Delta^*(\psi, \{false\}) = K\neg\psi$	$eval(\psi)(db_2) \in \{true, undef\}$
$\Delta^*(\psi, \{undef\}) = \neg K\psi \wedge \neg K\neg\psi$	$eval(\psi)(db_2) \in \{true, false\}$
$\Delta^*(\psi, \{true, false\}) = K\psi \;\vee\; K\neg\psi$	$eval(\psi)(db_2) = undef$
$\Delta^*(\psi, \{true, undef\}) = K\psi \;\vee\; \neg K\psi \wedge \neg K\neg\psi$	$eval(\psi)(db_2) = false$
$\Delta^*(\psi, \{false, undef\}) = K\neg\psi \;\vee\; \neg K\psi \wedge \neg K\neg\psi$	$eval(\psi)(db_2) = true$

The latter condition corresponds to condition (b) of Definition 11: Given a potential secret $\Psi = \Delta^*(\psi, V)$, there exists an indistinguishable instance db_2 with $eval(\psi)(db_2) \notin V$.

Lemma 15. *The combined lying and refusal method $cqe_{combined}$ presented in Section 2 is adapted for epistemic potential secrets.*

Proof. Condition 1: $cqe_{combined}$ only considers the potential secrets when determining the security configuration C_i (3). The implication operator $\models_{S5}$ employed allows epistemic sentences on its right hand side.

Condition 2: Consider the construction of db_2 (4) in the proof sketch of Theorem 5. Let ψ be a propositional sentence. We investigate the six ways to construct an epistemic potential secret Ψ from ψ, according to Definition 14.

Let $M = (S, \mathcal{K}, \pi)$ an $\mathcal{M}_{DS}$-structure and $s \in S$ a state such that $(M, s) \models log_n$ but $(M, s) \not\models \Psi$.

Case 1 ($\Psi = K\psi$).

Then we have $(M, s) \not\models K\psi$ and, according to (4), $\psi \notin db_2$. As db_2 is closed under logical implication, we conclude that $db_2 \not\models_{PL} \psi$ and thereby $eval(\psi)(db_2) \in \{false, undef\}$.

Case 2 ($\Psi = K\neg\psi$).

Similar to Case 1 (consider $\psi' = \neg\psi$).

Case 3 ($\Psi = \neg K\psi \wedge \neg K\neg\psi$).

Then we have $(M, s) \not\models \neg K\psi \wedge \neg K\neg\psi$, which means that $(M, s) \models K\psi \vee K\neg\psi$ and thus either $(M, s) \models K\psi$ or $(M, s) \models K\neg\psi$. Hence, it holds that either $\psi \in db_2$ or $\neg\psi \in db_2$. By the closure of db_2 and the definition of the *eval* function (1), we then have $eval(\psi)(db_2) \in \{true, false\}$.

Case 4 ($\Psi = K\psi \;\vee\; K\neg\psi$).

Then it holds that $(M, s) \not\models K\psi \vee K\neg\psi$, which means that $(M, s) \not\models K\psi$ and $(M, s) \not\models K\neg\psi$. By the results from Case 1 and Case 2, we then have $eval(\psi)(db_2) \in \{false, undef\}$ and $eval(\psi)(db_2) \in \{true, undef\}$, so it must hold that $eval(\psi)(db_2) = undef$.

Case 5 ($\Psi = K\psi \ \vee \ \neg K\psi \wedge K\neg\psi$).
Accordingly.
Case 6 ($\Psi = \neg K\psi \ \vee \ \neg K\psi \wedge K\neg\psi$).
Accordingly.

Given an enforcement method adapted for epistemic potential secrets, we can employ the reduction outlined above. The confidentiality targets are converted into potential secrets and then passed to the underlying enforcement method. The established enforcement method then satisfies our notion of confidentiality.

Theorem 16. *Let cqe be an enforcement method for potential secrets, preserving confidentiality (Definition 4) and being adapted for epistemic potential secrets (Definition 14). Let precondition be the associated precondition. Then the enforcement method for confidentiality targets given by the function*

$$cqe^*(Q, db, prior, policy) := cqe(Q, db, prior, pot\_sec(policy))$$

with the precondition

$$precondition^*(db, prior, policy) := precondition(db, prior, pot\_sec(policy))$$

preserves confidentiality in the sense of Definition 11.

Proof. Let db_1 be a database instance, *policy* a generalized confidentiality policy, *prior* the a priori assumptions so that the pertinent $precondition(db_1, log_0, pot\_sec(policy))$ is satisfied, and $Q = \langle \Phi_1, \ldots, \Phi_n \rangle$ a query sequence.

Let $(\psi, V) \in policy$ be a confidentiality target. By the definition of the *pot_sec* function (5), $pot\_sec(policy)$ contains the potential secret $\Psi = \Delta^*(\psi, V)$. By condition (2) of Definition 14, there exists a database instance db_2 under which the same answers and log files are generated as under db_1, and with $eval(\psi)(db_2) \notin V$. This satisfies conditions (a) and (b) of Definition 11.

5 Conclusion

In incomplete information systems, a sentence can be protected in various semantic ways. In particular, definite and partial information about a truth value of some sentence can be protected, or a combination thereof. We specified a generalized framework for expressing such a confidentiality target as a pair of a propositional sentence and a set of "forbidden" query values. We then gave a formal definition of confidentiality which resembles the respective notions for both potential secrets and secrecies under complete information systems and potential secrets for incomplete information systems. Instead of designing specific dedicated enforcement methods for each semantic type of confidentiality target, we picked up the idea of *naive reduction* [18] and showed how to convert a generalized confidentiality policy into a set of epistemic potential secrets, which can be used as the input to the existing enforcement methods for potential secrets. As these existing methods are originally designed for propositional potential secrets, we had to prove that the methods are *adapted for epistemic potential secrets.*

Alternatively, a security administrator may decide to specify the confidentiality policy as a set of epistemic potential secrets in the first place, given that these epistemic sentences have one of the six syntactical forms given in Definition 14. We however believe that it is favorable to specify the confidentiality policy with the means of confidentiality targets, for the sake of easier administration.

Although confidentiality targets provide a higher expressiveness than ordinary (propositional) potential secrets, there are still some limitations, in particular when you want to protect information about two different propositional sentences at the same time. For example, the information "a is *true* and b is (at the same time) *undef*" cannot be formalized as a confidentiality target (as there is no conjunction operator for confidentiality targets, and also no disjunction or negation). It is however easy to express this information as an epistemic sentence: $Ka \wedge \neg Kb \wedge \neg K \neg b$. In order to protect such a sentence, we would need to extend our framework such that it can handle a wider variety of epistemic sentences as potential secrets (essentially those in which any propositional sub-formula is prefixed by K). This will be the topic of future work.

A prototype implementation of the work presented in this paper is available from [23]. With these results, six of the twelve scenarios for complete information systems (resulting from the three parameters: potential secrets/secrecies, known/unknown policy, lying/refusal/combined lying and refusal) have been translated to incomplete databases. Current work includes the investigation of *unknown* policies, and how to exploit the situation when the user does not know which sentences are protected. The results for complete databases [19] suggest that less answers need to be distorted then.

At the moment, our work is limited to closed (yes/no-)queries and propositional logic. A useful application might be, for example, trust negotiation [24], a technique to establish trust between two agents by subsequently presenting credentials to each other. CQE could assist the agents to protect sensitive information while exchanging their credentials. This will be covered by future work. Considerations about open queries and first-order logic can be found in [16]; we are currently working on an implementation that will act as an interface layer to the Oracle DBMS. We also plan to investigate how to handle updates to the database instance, and how to deal with the situation when the log file contains information that has become obsolete due to a modified instance.

References

1. Castano, S., Fugini, M., Martella, G., Samarati, P.: Database Security. ACM Press, New York (1995)
2. Denning, D.: Cryptography and Data Security. Addison-Wesley, London, UK (1982)
3. Leiss, E.L.: Principles of Data Security. Plenum Press, New York (1982)
4. Domingo-Ferrer, J. (ed.): Inference Control in Statistical Databases. In: Domingo-Ferrer, J. (ed.) Inference Control in Statistical Databases. LNCS, vol. 2316, Springer, Heidelberg (2002)

5. Wang, L., Jajodia, S., Wijesekera, D.: Securing OLAP data cubes against privacy breaches. In: IEEE Symposium on Security and Privacy, pp. 161–178. IEEE Computer Society, Los Alamitos (2004)
6. Wang, L., Li, Y., Wijesekera, D., Jajodia, S.: Precisely answering multi-dimensional range queries without privacy breaches. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, Springer, Heidelberg (2003)
7. Brodsky, A., Farkas, C., Jajodia, S.: Secure databases: Constraints, inference channels, and monitoring disclosures. IEEE Transactions on Knowledge and Data Engineering 12(6), 900–919 (2000)
8. Lunt, T.F., Denning, D.E., Schell, R.R., Heckman, M., Shockley, W.R.: The seaview security model. IEEE Transactions on Software Engineering 16(6), 593–607 (1990)
9. Qian, X., Lunt, T.F.: A semantic framework of the multilevel secure relational model. IEEE Transactions on Knowledge and Data Engineering 9(2), 292–301 (1997)
10. Staddon, J.: Dynamic inference control. In: 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 94–100 (2003)
11. Winslett, M., Smith, K., Qian, X.: Formal query languages for secure relational databases. ACM Transactions on Database Systems 19(4), 626–662 (1994)
12. Farkas, C., Jajodia, S.: The inference problem: A survey. SIGKDD Explorations 4(2), 6–11 (2002)
13. Sicherman, G.L., de Jonge, W., van de Riet, R.P.: Answering queries without revealing secrets. ACM Transactions on Database Systems 8(1), 41–59 (1983)
14. Bonatti, P.A., Kraus, S., Subrahmanian, V.: Foundations of secure deductive databases. IEEE Transactions on Knowledge and Data Engineering 7(3), 406–422 (1995)
15. Biskup, J.: For unknown secrecies refusal is better than lying. Data & Knowledge Engineering 33, 1–23 (2000)
16. Biskup, J., Bonatti, P.A.: Controlled query evaluation with open queries for a decidable relational submodel. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 43–62. Springer, Heidelberg (2006)
17. Biskup, J., Bonatti, P.A.: Lying versus refusal for known potential secrets. Data & Knowledge Engineering 38, 199–222 (2001)
18. Biskup, J., Bonatti, P.A.: Controlled query evaluation for enforcing confidentiality in complete information systems. International Journal of Information Security 3, 14–27 (2004)
19. Biskup, J., Bonatti, P.A.: Controlled query evaluation for known policies by combining lying and refusal. Annals of Mathematics and Artificial Intelligence 40, 37–62 (2004)
20. Biskup, J., Weibert, T.: Refusal in incomplete databases. In: Research Directions in Data and Applications Security XVIII, pp. 143–157 Kluwer/Springer (2004)
21. Biskup, J., Weibert, T.: Keeping secrets in incomplete databases. Submitted, 2007. Extended abstract presented at the LICS'05 Affiliated Workshop on Foundations of Computer Security (FCS'05), available from http://www.cs.chalmers.se/ andrei/FCS05/fcs05.pdf(2005)
22. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. MIT Press, Cambridge (1995)
23. University of Dortmund, Information Systems and Security: CQE prototype implementation. http://ls6-www.cs.uni-dortmund.de/issi/projects/cqe/
24. Winslett, M.: An introduction to trust negotiation. In: Nixon, P., Terzis, S. (eds.) iTrust 2003. LNCS, vol. 2692, pp. 275–283. Springer, Heidelberg (2003)

Provably-Secure Schemes for Basic Query Support in Outsourced Databases

Georgios Amanatidis, Alexandra Boldyreva, and Adam O'Neill

Georgia Institute of Technology, USA
amana@math.gatech.edu, {aboldyre,amoneill}@cc.gatech.edu

Abstract. In this paper, we take a closer look at the security of outsourced databases (aka Database-as-the-Service or DAS), a topic of emerging importance. DAS allows users to store sensitive data on a remote, untrusted server and retrieve desired parts of it on request. At first we focus on basic, exact-match query functionality, and then extend our treatment to prefix-matching and, to a more limited extent, range queries as well. We propose several searchable encryption schemes that are not only practical enough for use in DAS in terms of query-processing efficiency but also provably-provide privacy and authenticity of data under new definitions of security that we introduce. The schemes are easy to implement and are based on standard cryptographic primitives such as block ciphers, symmetric encryption schemes, and message authentication codes. As we are some of the first to apply the provable-security framework of modern cryptography to this context, we believe our work will help to properly analyze future schemes and facilitate further research on the subject in general.

1 Introduction

MOTIVATION. Outsourcing data to off-site database service providers is becoming an attractive, cost-effective option for many organizations [40]. In this setting (also known as Database-as-a-Service or DAS), a client stores data on a remote untrusted database server and queries the server in order to receive required portions of the data. Usually this data is stored in the form of a relational database, each divided into records (or tuples) with attributes (or fields). The basic system requirements are (1) query support, (2) computation and communication efficiency for both client and server, and (3) data security. Note that the latter requirement is particularly important in DAS, as data often contains sensitive financial, medical, or intellectual information and the server cannot be trusted. Indeed, ensuring security in DAS is an important research topic that has been receiving increasing attention [25,37,27,26,20,21,37,3,28,2,29,31,9], and security may even be required by law (cf. HIPAA rules [1]).

The problem is that these requirements are in conflict with each other. For example, consider encrypting the data with a secure encryption scheme that hides *all* information and is always randomized (i.e. same messages yields completely different ciphertexts). This does not allow the user to even form a query about any set of records smaller than the the whole database. Indeed, it turns out that even addressing just the basic exact-match (point) queries is a non-trivial task if one wants to treat security in a systematic, not ad-hoc, way.

Previous Work. Searching on encrypted data has been a topic of multiple relevant works in the cryptographic community, which focus mainly on exact-match queries but in an unsatisfactory way for our context. In particular, the schemes of [41,22,24,15,18,19] provide strong security guarantees (typically revealing only the user access pattern) while allowing a server to answer exact-match queries, but doing so *requires the server to scan the whole database for each query*, yielding unacceptably-slow performance for medium-size to very large databases. The schemes of [19] get around this problem by requiring the (paying) client to know all keywords and all data beforehand and pre-computing a static index for the server that does not allow to treat relational databases. A fundamental question thus becomes what is the best guaranteed security that can be achieved without compromising general efficiency and functionality. The work of [9] recently raised this problem in the asymmetric (public-key) setting, where users explicitly consist of "senders" and "receivers," and provided new security definitions and provably-secure solutions for exact-match queries. We consider this problem entirely in the more-common symmetric-key setting where a client (which may be a large group of users, e.g. in a business) both stores and queries its own data on an untrusted server.

Research on this subject done in the database community focuses on the first two requirements and provides encryption schemes with attractive functionality, namely efficient and optimized indexing and flexible query support e.g. for numerical range, comparison, or aggregation queries [37,3,25,21,27,28,26,31]. In contrast, the security of these schemes is far less clear. Many utilize cryptographic primitives, such as order-preserving hash functions and encryption schemes, which have not been studied by cryptographers, and without scrutinizing their security. For example, using a deterministic encryption scheme for point queries sounds like a reasonable idea, because then forming a point query is feasible and the server can efficiently index and locate the ciphertexts. But what scheme should be used? One common suggestion (see e.g. [28,2]) is to use DES or AES. But these are block ciphers for short plaintexts of at most 128 bits. If a database field holds larger data, say barcode information, then it is not clear how to encrypt longer ones. It would be natural to apply the block cipher block-by-block, but then the adversary will see when the underlying plaintexts have common blocks, which is an unnecessary leak of information. Similarly, fixing the randomness in an arbitrary encryption mode (e.g. CBC) will leak more information than needed.

A noteworthy exception in this body of work is a recent paper by M. Kantarcioglu and C. Clifton [29], which calls for a new direction of research that aims for "efficient encrypted database and query processing with *provable* security properties." Their work provides a first step in this direction. As they observe, unless one lowers the security bar from the previous cryptographic solutions a linear scan of the database on each query is fundamentally necessary. But the above discussion suggests we must be careful to not go too far. On the other hand, the security definition proposed in [29] requires the use of server-side trusted, tamper-resistant hardware to achieve.

Overview of Contributions. In a broad sense, our goal is to narrow the gap between query-processing-efficient but ad-hoc schemes with unclear security and schemes with strong security guarantees but with unsuitable functionality. We review the provable-security methodology in Section 2. Then to start with, we consider exact-match queries (i.e. with boolean conditions involving only equalities). In Section 4, we formulate what algorithms and properties constitute an *efficiently-searchable authenticated encryption or ESAE* scheme that will allow a server to process such queries, when used to separately encrypt each searchable field, with, unlike for previous cryptographic-community schemes, query-processing efficiency comparable to that for unencrypted databases.

As opposed to previous works in the database community, we go significantly beyond explaining why some attacks do or do not work in order to develop a *foundation* for our understanding of security. Observe that while typically encryption hides all partial information about the data (which is still true for previous searchable schemes in the cryptographic community, and homomorphic encryption schemes in a basic model of security), ESAE cannot because some information needs to be leaked to allow efficient query processing. Hence we formulate a new definition of security that captures the intuition that no adversary should be able to learn *any* useful information about the data within reasonable time, beyond what is unavoidable for the given functionality, namely when two ciphertexts correpond to equal plaintexts; we argue that permitting false-positive results cannot help to hide this correlation in practice. Our definition moreover captures a notion of authenticity that ensures attributes values are not modified or added over the network or at the server side without the user noticing.[1] Thus in a sense we provide the strongest possible notion of security one can reasonably ask for without relying on trusted hardware as in [29]. Note that we do not explicitly model security in the terms of a client-database interaction but always instead simply derive security in this context from that of the "ideal" cryptographic object in question. (This step is crucially absent in [31].) In Section 5 we propose and analyze two exact-match ESAE constructions meeting our definition.

Then in Section 6 we extend our framework to treat prefix-matching queries as well and refer to [4], where we investigate a recent approach [31] to handling range queries and point out some difficulties in achieving a reasonable level of security with it.

[1] The issues of authenticity for the database and the records as a whole, and ensuring that the server returns all the current, requested data, are outside our scope and can be dealt with the methods of [32,35,36,30].

2 The Provable-Security Methodology

Cryptographic protocols were often designed by trial-and-error, where a scheme is implemented and used until some flaws are found and fixed, if possible, and the revised scheme is used until new flaws are found, and so forth. A revolutionary and superior "provable-security" approach was originally proposed by Goldwasser and Micali [23]. The approach requires a formal definition of a security goal (e.g., data privacy) for a given cryptographic object (e.g., an encryption scheme). A security definition comprises a formal description of adversarial capabilities (what an adversary knows and can do) and of what an adversary must do to break the scheme. A proof of security then shows by reduction that a given scheme satisfies the definition under widely accepted assumptions (e.g., that factoring big composite numbers or distinguishing outputs of a block cipher from random strings is hard). The proof thus shows that *the only way* to break the scheme in reasonable time is by breaking the underlying assumption about the hard problem. See [6] for a detailed overview of the provable-security framework.

3 Preliminaries

Notation. We refer to members of $\{0,1\}^*$ as strings. If X is a string then $|X|$ denotes its length in bits and if X, Y are strings then $X \| Y$ denotes the concatenation of X and Y. If S is a set then $X \overset{\$}{\leftarrow} S$ denotes that X is selected uniformly at random from S. If A is a randomized algorithm then $A(x, y, \ldots; R)$, or $A(x, y, \ldots)$ for short, denotes the result of running A on inputs $x, y, \ldots$ and with coins R, and $a \overset{\$}{\leftarrow} A(x, y, \ldots)$ means that we choose R at random and let $a = A(x, y, \ldots; R)$. Oracle access, when given to algorithms (and denoted by superscript), is done as a "black-box," meaning the algorithms see only the input slots provided to them.

Symmetric encryption and message authentication. We recall the basics concerning symmetric encryption and, following this, message authentication.

Definition 1. [Symmetric encryption] *A symmetric encryption scheme* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *with associated message space* $\mathsf{MsgSp}(\mathcal{SE})$ *consists of three algorithms. (1) The randomized key generation algorithm* $\mathcal{K}$ *returns a secret key sk; we write sk* $\overset{\$}{\leftarrow} \mathcal{K}$*. (2) The (possibly randomized) encryption algorithm* $\mathcal{E}$ *takes input the secret key sk and a plaintext* m *to return a ciphertext; we write* $C \overset{\$}{\leftarrow} \mathcal{E}(sk, m)$ *or* $C \leftarrow \mathcal{E}(sk, m; R)$*. If* $C = \mathcal{E}(sk, m, R)$ *for some coins* R *then we say* C *is a* valid *ciphertext for* m *under sk. (3) The deterministic decryption algorithm* $\mathcal{D}$ *takes the secret key sk and a ciphertext* C *to return the corresponding plaintext or a special symbol* $\perp$ *indicating that the ciphertext was invalid; we write* $m \leftarrow \mathcal{D}(sk, C)$ *(or* $\perp \leftarrow \mathcal{D}(sk, C)$*.)*

Consistency: we require that $\mathcal{D}(sk, (\mathcal{E}(sk, m)) = m$ *for all* $m \in \mathsf{MsgSp}(\mathcal{SE})$*.*

The idea behind security of encryption is that an adversary against a scheme should not be able to deduce anything about the underlying message (except its length, which encryption cannot hide), upon seeing the ciphertext, even if it has some *a priori* information of its choice about the message. This intuition is captured via a notion of "indistinguishability" of encryptions [11], which requires that no efficient adversary should be able to distinguish between encryptions of two messages, even if the adversary can choose these two messages and request to see ciphertexts of other different messages of its choice.

Definition 2. [Security of encryption] *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme with* $\mathsf{MsgSp}(\mathcal{SE})$*. Let $\mathcal{LR}$ (left-or-right) be the "selector" that on input m_0, m_1, b returns m_b. The scheme $\mathcal{SE}$ is said to be* secure against chosen-plaintext attack *or* ind-cpa *if for every efficient adversary B the value called the advantage of B $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},B}$ is sufficiently small, where*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},B} = \Pr[\,\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{SE},B} = 0\,] - \Pr[\,\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{SE},B} = 0\,]$$

and the experiments above are defined for $b \in \{0,1\}$ and an ind-cpa adversary B who is required to query messages of equal length and in $\mathsf{MsgSp}(\mathcal{SE})$*, as:*

$$\textbf{\textit{Experiment}}\ \mathbf{Exp}^{\text{ind-cpa-}b}_{\mathcal{SE},B}$$
$$sk \xleftarrow{\$} \mathcal{K}\ ;\ d \xleftarrow{\$} B^{\mathcal{E}(sk,\mathcal{LR}(\cdot,\cdot,b))}\ ;\ \textit{Return } d$$

∎

We purposely do not mathematically define an "efficient" adversary and how "small" the advantage should be. This will vary according to the particular appliation. For example, guaranteeing that all adversaries whose running time is up to 2^{60} in some fixed RAM model of computation have maximum advantage 2^{-20} would usually be considered sufficient.

Definition 3. [MAC] *A deterministic message authentication code or MAC scheme $\mathcal{MAC} = (\mathcal{K}, \mathcal{M}, \mathcal{V})$ with associated message space* $\mathsf{MsgSp}(\mathcal{MAC})$ *consists of three algorithms. (1) The randomized key generation algorithm $\mathcal{K}$ returns a a secret key sk; we write $sk \xleftarrow{\$} \mathcal{K}$. (2) The deterministic mac algorithm $\mathcal{M}$ takes input the secret key sk and a plaintext m to return a "mac" for m; we write $\sigma \leftarrow \mathcal{M}(sk, m)$.(3) The deterministic verification algorithm $\mathcal{V}$ takes the secret key sk, a message m, and a mac σ to return a bit $b \in \{0,1\}$; we write $b \leftarrow \mathcal{V}(sk, m, \sigma)$. If b is 1 we say that σ is a valid mac for m under sk.*

Consistency: we require that $\mathcal{V}(sk, m, (\mathcal{M}(sk, m)) = 1$ for all $m \in$ MsgSp *$(\mathcal{MAC})$.*

More generally, one can permit $\mathcal{M}$ to flip coins as well, but most practical MACs (e.g., CMAC or HMAC) are deterministic, which is important in our context. Thus in this paper "MAC" means "deterministic MAC."

The standard definition of security of MACs, unforgeability under chosen-message attacks (or uf-cma) requires that no efficient adversary that sees macs of the messages of its choice can produce a valid mac for a new message.

Definition 4. [Security of MACs] *A MAC scheme $\mathcal{MAC} = (\mathcal{K}, \mathcal{M}, \mathcal{V})$ is said to be unforgeable against chosen-message attack or uf-cma if for every efficient adversary B the value $\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MAC},B}$ called* advantage *of B is sufficiently small, where*

$$\mathbf{Adv}^{\text{uf-cma}}_{\mathcal{MAC},B} = \Pr[\,\mathbf{Exp}^{\text{uf-cma}}_{\mathcal{MAC},B} = 1\,] \quad \textit{and the experiment is defined as}$$

$$\textbf{\textit{Experiment}}\ \mathbf{Exp}^{\text{uf-cma}}_{\mathcal{MAC},B}$$
$$sk \xleftarrow{\$} \mathcal{K}\,;\ (m, \sigma) \xleftarrow{\$} B^{\mathcal{M}(sk,\cdot),\mathcal{V}(sk,\cdot,\cdot)}\,;\ \textit{Return}\ \mathcal{V}(sk, m, \sigma)$$

and B is not allowed to query m to its mac oracle. ▮

We will also use an additional property of MACs, namely privacy preservation, originating recently in [12], which requires the outputs of the MAC to hide information about the messages similarly to encryption.

Definition 5. [Privacy-preserving MACs] *[7,12] A MAC scheme $\mathcal{MAC} = (\mathcal{K}, \mathcal{M}, \mathcal{V})$ is said to be* privacy-preserving *if for every efficient adversary B the value called the advantage of B $\mathbf{Adv}^{\text{pp-mac}}_{\mathcal{MAC},B}$ is sufficiently small, where*

$$\mathbf{Adv}^{\text{pp-mac}}_{\mathcal{MAC},B} = \Pr[\,\mathbf{Exp}^{\text{pp-mac-0}}_{\mathcal{MAC},B} = 0\,] - \Pr[\,\mathbf{Exp}^{\text{pp-mac-1}}_{\mathcal{MAC},B} = 0\,]$$

and the experiments above are defined for the adversary B and $, b \in \{0, 1\}$ as

$$\textbf{\textit{Experiment}}\ \mathbf{Exp}^{\text{pp-mac-}b}_{\mathcal{MAC},B}$$
$$sk \xleftarrow{\$} \mathcal{K}\,;\ d \xleftarrow{\$} B^{\mathcal{M}(sk,\mathcal{LR}(\cdot,\cdot,b))}\,;\ \textit{Return}\ d$$

Above $\mathcal{LR}$ is the oracle that on input m_0, m_1, b returns m_b; and we require that for any sequence of oracle queries $(m_{1,1}, m_{1,2}), \ldots, (m_{q,1}, m_{q,2})$ that B can make to its oracle, there does not exist any $m_{i,1} = m_{j,1}$ or $m_{i,2} = m_{j,2}$ for $i \neq j$ and moreover $|m_{i,1}| = |m_{i,2}|$ for all i. ▮

4 Efficiently-Searchable Authenticated Encryption

What is ESAE. We now define the syntax of an ESAE (Efficiently-Searchable Authenticated Encryption) scheme.

Definition 6. [ESAE] *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme. We say that $\mathcal{ESAE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{F}, \mathcal{G})$ an* efficiently-searchable authenticated encryption *(*ESAE*) scheme if $\mathcal{K}, \mathcal{E}, \mathcal{D}$ are the algorithms of a regular encryption scheme and $\mathcal{F}, \mathcal{G}$, are deterministic efficient algorithms where the former takes a secret key and message as input and the latter takes a ciphertext and:*
(1) Completeness:

$$\Pr\Big[\, sk \xleftarrow{\$} \mathcal{K}\,;\ f_1 \leftarrow \mathcal{F}(sk, m_1)\,;\ g_1 \leftarrow \mathcal{G}(\mathcal{E}(sk, m_1)) \ :\ f_1 = g_1 \Big] = 1 \quad \textit{and}$$

(2) Soundness:

$$\Pr\Big[\, sk \xleftarrow{\$} \mathcal{K}\,;\ (m_0, m_1) \xleftarrow{\$} \mathcal{M}_{\mathcal{SE}} \ :\ \mathcal{F}(sk, m_0) = \mathcal{G}(\mathcal{E}(sk, m_1)) \Big]\ \textit{is sufficiently}$$

small

for every message $m_1 \in \mathsf{MsgSp}(\mathcal{SE})$ and every efficient randomized algorithm $\mathcal{M}_{\mathcal{SE}}$ that outputs distinct messages $m_0, m_1 \in \mathsf{MsgSp}(\mathcal{SE})$. We refer to the output of $\mathcal{F}, \mathcal{G}$ as the tag *of a message m or a corresponding ciphertext C.*

The algorithm $\mathcal{F}$ is used by the user to form queries, and $\mathcal{G}$ is needed by the server to be able to index the encrypted data *a priori*, using the standard data structures (e.g. B-tress), and locate records on request (see below), for which it is crucial that $\mathcal{F}, \mathcal{G}$ are not randomized. Thus the completeness property ensures that encrypted data can be efficiently searched, in logarithmic-time in the database size, meaning this time has not gone up over unencrypted data. The soundness property ensures that false positives do not occur too often so that post-processing is efficient. We first focus on the case that the soundness probability in the definition so small that each ciphertext essentially has a unique tag; we will address increasing the number false-positive results later.

Note that exact-match functionality can also be used to build various other useful more-complicated query types. These include equijoin and group-by, the latter of which is especially useful for example in supporting multi-faceted search that projects among various dimensions (e.g. features/types of products). Moreover, the server can *ipso facto* compute counts over the data, which would also be useful in this context for example to support a product search interface that shows there are, say, 100 CRT and 200 LCD monitors in the database, and 100 15", 100 17", and 100 20" monitors. You click on LCD monitors link and it now shows 50 15", 75 17", and 75 20" such monitors.

Security of ESAE. Efficient "searchability" (ensured by the completeness property) necessarily violates the standard ind-cpa security for encryption. Thus we provide a relaxed definition suitable for given functionality. Completeness implies that the server (and the adversary) will always be able to see what ciphertexts correspond to equal plaintexts, and a security definition should ensure that this is *all* the adversary can learn. To this end we design an indistinguishability experiment (cf. Definition 2) where we disallow the adversary from seeing ciphertexts of equal messages such that it can trivially succeed. The adversary can also mount chosen-ciphertext attacks according to a relaxed chosen-ciphertext-security definition [5,16] that is suitable for our application. For integrity of the data, we also want to require that it is hard produce a new ciphertext or change the existing one without the user noticing, which corresponds to a notion of ciphertext-integrity for authenticated encryption [13].

Definition 7. [Security of ESAE] *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{F}, \mathcal{G})$ be an ESAE scheme. Let $\mathcal{LR}$ (left-or-right) be the selector that on input m_0, m_1, b returns m_b. Let B be an adversary who is given access to two oracles (called lr-encryption and the decryption oracles). For $b \in \{0,1\}$ define the experiment:*

```
Experiment $\mathbf{Exp}^{\mathrm{ind\text{-}esae\text{-}}b}_{\mathcal{SE},B}$
    $sk \xleftarrow{\$} \mathcal{K}$ ; $d \xleftarrow{\$} B^{\mathcal{E}(sk,\mathcal{LR}(\cdot,\cdot,b)),\mathcal{D}(sk,\cdot)}$
    If $m \neq \perp$ was returned from $\mathcal{D}(sk,\cdot)$ at any point then $d \leftarrow b$
    Return $d$
```

We call B an esae adversary *if for any sequence of queries* $(m_{1,1}, m_{1,2}), \ldots, (m_{q,1}, m_{q,2})$ *that B can make to its lr-encryption oracle, there does not exist any* $m_{i,1} = m_{j,1}$ *or* $m_{k,2} = m_{l,2}$ *for* $i \neq j, k \neq l$ *such that* $m_{i,2} \neq m_{j,2}$ *or* $m_{k,1} \neq m_{l,1}$*, in addition to the usual requirements that* $|m_{i,1}| = |m_{i,2}|$ *for all i and if B does not query the decryption oracle on a ciphertext that has the same tag as any ciphertext that has been returned by the lr-encryption oracle. The* advantage *of an esae adversary B is defined as follows:*

$$\mathbf{Adv}^{\text{ind-esae}}_{\mathcal{SE},B} = \Pr[\,\mathbf{Exp}^{\text{ind-esae-0}}_{\mathcal{SE},B} = 0\,] - \Pr[\,\mathbf{Exp}^{\text{ind-esae-1}}_{\mathcal{SE},B} = 0\,].$$

The ESAE scheme $\mathcal{SE}$ is said to be esae-secure *if for every efficient esae adversary B the function* $\mathbf{Adv}^{\text{ind-esae}}_{\mathcal{SE},B}$ *is sufficiently small.* ▮

We note the similarity of ESAE to deterministic authenticated encryption (DAE), studied in [39] in the context of transporting (encrypted) symmetric keys. However, the definition of security for DAE in [39] is shown there be equivalent to that for "pseudorandom injections," and we will see that an ESAE scheme need not be pseudorandom nor deterministic.

Discussion. In the context of DAS, the server receives queries with tags for the data, the former of which it would have computed itself, thus the definition of security we provide essentially guarantees that the server cannot learn anything about the data of the user beyond its occurrence profile (or distribution), i.e. how many times a given attribute value (without knowing anything else about it) occurs in the database and in which records, even if it is one of only two possible such values that it can pick itself, and analogously the user access pattern.

As for authenticity (aka. integrity) of ciphertexts, our definition guarantees integrity in that any modification or substitution (malicious or not) to the encrypted data is detected by the user. We note that authenticity is ensured at the field level, and not on the record level or for the entire database; an adversary can still, for example, switch (encrypted) attribute values stored in different records. If the data is updated and returned as whole records, then one can simply authenticate at the record level instead. In many applications, the server can be trusted to return the correct ciphertexts to its paying customers (even when it may try to learn and sell their data). Thus one should mainly protect against non-adversarial transmission or storage errors, and our definition does it.

Increased False-Positives. It seems intuitive that permitting false positive results (i.e. relaxing the soundness condition in Definition 6) via a "bucketization" technique where a fixed number of randomly-chosen plaintexts correspond to each tag, [34,33,17], though requiring the client to do more work to filter out these false-positives, would allow a proportional increase in security by preventing the adversary from correlating equal plaintexts. But we claim that this intuition is not always correct; in practice such information may still be leaked. To see this, consider the *a posteriori* probability of a plaintext occurring a certain number of times given an occurrence distribution on the buckets; the "farther" the latter is from the uniform distribution means a better estimate on the plaintext occurrence profile, and one cannot expect anything close to the uniform distribution in practice. One solution would be make the bucket distribution instead depend on that of the input, but in particular as noted in [34] this would require impractical communication cost between client and server as this distrbution changes over time, and it is noted in [31] that such mappings are typically not efficiently computable, making storing and managing them impractical.

Comparison to the model from [29] . The security definition of [29] guarantees that an adversary (e.g. the server) cannot distinguish between two queries whose results sets have the same size, whereas ours reveals which records are accessed by such queries. This hold even with respect to extremely powerful adversaries who can mount chosen-ciphertext attacks, whereas our definition applies to somewhat more passive adversaries, which we nevertheless believe is reasonable for the given application. On the other hand, the definition of [29] requires server-side trusted hardware to achieve.

5 Proposed Constructions and Their Security Analyses

Mac-and-encrypt. We first present an easy-to-implement, "off the shelf" way to construct an ESAE scheme from any encryption and MAC schemes and then analyze its security and comment on implementation.

Definition 8. [Mac-and-encrypt construction] *Let $\mathcal{SE} = (\mathcal{K}_E, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and $\mathcal{MAC} = (\mathcal{K}_M, \mathcal{M}, \mathcal{V})$ be a message authentication code. Then we define a new symmetric encryption scheme $\mathcal{SE}^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*, \mathcal{F}, \mathcal{G})$, whose constituent algorithms work as follows:*

- *$\mathcal{K}^*$ sets $sk_M \overset{\$}{\leftarrow} \mathcal{K}_M$ and $sk_E \overset{\$}{\leftarrow} \mathcal{K}_E$, then outputs $sk_M \| sk_E$.*
- *$\mathcal{E}^*$ on input $sk_M \| sk_E, m$, sets $\sigma \leftarrow \mathcal{M}(sk_M, m)$ and $C \overset{\$}{\leftarrow} \mathcal{E}(sk_E, m)$, then outputs $\sigma \| C$.*
- *$\mathcal{D}^*$ on input $sk_M \| sk_E, \sigma \| C$, first sets $m \leftarrow \mathcal{D}(sk_E, C)$ and then $b \leftarrow \mathcal{V}(sk_M, m, \sigma)$. It outputs m if $b = 1$ and $\perp$ otherwise.*
- *$\mathcal{F}$ and $\mathcal{G}$ on inputs $sk_M \| sk_E, m$ and $\sigma \| C$, respectively, return $\mathcal{M}(sk_M, m)$ and σ.*

We first argue that $\mathcal{SE}^*$ is an ESAE scheme if $\mathcal{MAC}$ is uf-cma. Clearly completeness is satisfied. The soundness condition relies on the uf-cma security of $\mathcal{MAC}$. Namely, suppose $\mathcal{MAC}$ is uf-cma but there is an algorithm $\mathcal{M}_{\mathcal{SE}}$ that outputs m_0, m_1 such that $\mathcal{M}(sk_M, m_0) = \mathcal{M}(sk_M, m_1)$ with high probability. This violates uf-cma security as follows. We construct a uf-cma adversary B as per Definition 4 that first runs $\mathcal{M}_{\mathcal{SE}}$ to receive its output (m_0, m_1) then queries its signing oracle for $\mathcal{M}(sk, m_0)$ to get back σ, and finally itself returns (m_1, σ). By the forgoing assumption on $\mathcal{M}_{\mathcal{SE}}$ this adversary has high uf-cma advantage, a contradiction.

Theorem 1. *Let $\mathcal{SE} = (\mathcal{K}_E, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and $\mathcal{MAC} = (\mathcal{K}_M, \mathcal{M}, \mathcal{V})$ be a deterministic MAC. Then let $\mathcal{SE}^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*, \mathcal{F}, \mathcal{G})$ be the* *mac-and-encrypt ESAE scheme defined according to Definition 8. We have that $\mathcal{SE}^*$ is esae-secure if $\mathcal{SE}$ is ind-cpa and $\mathcal{MAC}$ is uf-cma and privacy-preserving.*

Due to lack of space the concrete security statement that shows explicit relations between the advantages of the adversaries and the proof are given in [4].

There are many efficient and standardized provably-secure symmetric encryption and MAC schemes that can be used to build an ESAE scheme according to Definition 8. Our recommendations for encryption schemes include CBC and CTR (aka the counter or XOR) encryption modes based on the AES block cipher, which are proven to be ind-cpa under the assumption that AES is a pseudorandom function (PRF) [11]. For MACs, one can use SHA-1 or SHA-256 and AES-based HMAC or CMAC (a variation of CBC-MAC), proven uf-cma assuming the underlying hash function is collision-resistant or PRF and the block cipher is PRF [10,7,14]. Theorem 1 implies that the resulting mac-and-encrypt ESAE is secure under the respective assumptions.

We remark that in database literature (e.g. [25]), some proposed solutions for this problem suggest to use a "random one-to-one mapping" whose output is included with a ciphertext, in order to facilitate "searchability." Thus one interesting implication of the above result is that such a map need not be random, or even pseudorandom, in order to achieve the best-possible notion of security.

Encrypt-with-mac. We now present a construction that is more computation-efficient on the client side and more communication-efficient over the network. This can be crucial, for example, when users have a low-bandwidth connection to the database or are connecting via a battery-constrained device [35]. The idea is to use the mac of the plaintext "inside" the encryption, namely as the randomness used in the encryption algorithm of a standard encryption scheme.

Definition 9. [Encrypt-with-mac construction] *Let $\mathcal{SE} = (\mathcal{K}_E, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and $\mathcal{MAC} = (\mathcal{K}_M, \mathcal{M}, \mathcal{V})$ be a deterministic MAC. Then we define a new symmetric encryption scheme $\mathcal{SE}^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*, \mathcal{F}, \mathcal{G})$, whose constituent algorithms work as follows:*

- *$\mathcal{K}^*$ sets $sk_M \overset{\$}{\leftarrow} \mathcal{K}_M$ and $sk_E \overset{\$}{\leftarrow} \mathcal{K}_E$, then outputs $sk_M \| sk_E$.*
- *$\mathcal{E}^*$ on input $sk_M \| sk_E, m$, sets $\sigma \leftarrow \mathcal{M}(sk_M, m)$ and $C \leftarrow \mathcal{E}(sk_E, m; \sigma)$, then outputs C.*
- *$\mathcal{D}^*$ on input $sk_M \| sk_E, C$, first sets $m \leftarrow \mathcal{D}(sk_E, C)$. It outputs m if $C = \mathcal{E}(sk_E, m; \mathcal{M}(sk_M, m))$ and $\perp$ otherwise.*
- *$\mathcal{F}$ is same as $\mathcal{E}^*$. $\mathcal{G}$ on input C returns C.*

To see that $\mathcal{SE}^*$ is an ESAE scheme, we note that the completeness requirement is clearly satisfied and the probability in the soundness requirement is zero here due to the consistency requirement in Definition 1.

Ideally, we would like to prove that the above construction is ease-secure assuming that $\mathcal{MAC}$ is a uf-cma and $\mathcal{SE}^*$ is ind-cpa secure. However, slightly stronger assumptions turns out to be needed, but they are met by practical schemes anyway. First, we will need the mac algorithm of $\mathcal{MAC}$ to be a pseudorandom function (PRF). Naturally, this requires a mac to "look like random bits" without the secret key, a well-studied notion formalized as follows.

Definition 10. *A family of functions is a map $F\colon \{0,1\}^b \times \{0,1\}^c \to \{0,1\}^c$, where we regard $\{0,1\}^b$ as the* keyspace *for the function family in that a* key *$k \in \{0,1\}^b$ induces a particular function from this family, which we denote by $F(k,\cdot)$. The family F is said to be* pseudorandom *(or a PRF) if for every efficient adversary B given oracle access to a function, its prf-advantage*

$$\mathbf{Adv}^{\mathrm{prf}}_{F,B} = Pr\left[B^{F(k,\cdot)} = 0\right] - \Pr\left[B^{Q(\cdot)} = 0\right]$$

is sufficiently small, where $F(k,\cdot)$ is the oracle for a random instance of F (specified by a randomly chosen key k) and $Q(\cdot)$ is the oracle for a truly random function with the domain and range of $F(k,\cdot)$. Pseudorandom permutations (PRPs) are defined analogously, and in this case the adversary B above is also given an inversion oracle.

To define the assumption needed for encryption, let us say that an encryption scheme $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has a *max-collision probability* [9] $mc_{\mathcal{SE}}$ if we have that:

$$\Pr\left[\mathcal{E}(sk, m, R_1) = \mathcal{E}(sk, m, R_2)\right] \leq mc_{\mathcal{SE}} \ ,$$

for every $m \in \mathsf{MsgSp}(\mathcal{SE})$, where the probability is taken over the random choices of the key sk and coins R_1, R_2 (chosen independently).

All practical encryption schemes satisfy the above property. The proof of the following is in [4]. It also contains the concrete security statement. .

Theorem 2. *Let $\mathcal{SE} = (\mathcal{K}_E, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and $\mathcal{MAC}$ $= (\mathcal{K}_M, \mathcal{M}, \mathcal{V})$ be a deterministic MAC. Let $\mathcal{SE}^* = (\mathcal{K}^*, \mathcal{E}^*, \mathcal{D}^*)$ be the encrypt-with-mac ESAE scheme defined via Definition 9. Then $\mathcal{SE}^*$ is esae-secure if $\mathcal{MAC}$ is a PRF and $\mathcal{SE}$ is ind-cpa and has sufficiently small max-collision probability.*

The same recommendations for the underlying schemes (CBC, CTR modes, and HMAC and CMAC) we gave for the mac-and-encrypt construct apply here. As we mentioned, CBC and CTR are proven to be ind-cpa assuming the base block cipher is PRF. Randomized CBC and CTR have max-collision probability 2^{-128} when used with AES and the counter-based CTR has zero max-collision probability. HMAC was recently proved to be a PRF assuming the underlying hash function is PRF [7], and CMAC is known to be PRF assuming the base block cipher is PRF; Theorem 2 implies that the resulting encrypt-with-mac ESAE scheme is secure under these respective assumptions.

We remark that our construction is similar to the SIV ("synthetic initialization vector") construction for deterministic authenticated encryption (DAE) in [39]. Indeed, it is straightforward to check that a secure DAE scheme as defined in [39] is also secure as an ESAE scheme. However, our construction and analysis is in fact somewhat more general than the SIV construction, which pertains only to some "initialization-vector-based" symmetric encryption schemes (including CBC and CTR) that implicitly guarantee to meet the max-collision requirement we pinpoint for security.

6 Prefix-Preserving ESAE

Prefix-Matching Queries. We extend our ESAE framework to encryption that allows to efficiently process prefix-matching queries, i.e. locating records whose attribute value starts with a given prefix, for example all phone numbers starting with area-code 310.

Our treatment builds on the study of "online ciphers" (so-called because they can be used on streaming data without buffering) in [8], which we view here as deterministic length-preserving encryption schemes whose input is composed of fixed-length blocks (which we call "characters" of the prefixes), where the ith block of the output depends only on the first i blocks of the input. Thus if two plaintexts agree on their first k characters then so do their ciphertexts. Following Definition 4, to show this implies efficient prefix-searchability (via appropriate server-side index structures for the tuples) we make functions $\mathcal{F}, \mathcal{G}$ return the encryption of an l-character prefix and the first l characters of a ciphertext; the fact that completeness is one and soundness is zero follows from the fact that the encryption is deterministic.

In our construction, the characters of a prefix will be of the input-length for an underlying block cipher (e.g. 64 bits or 4 UTF-16 characters using DES-variants). At the cost of revealing more information to the server for a more flexible granularity of prefixes in the queries ,a *bitwise* prefix-preserving scheme of Xu et al. [42] can similarly be used here (an issue we will return to later), which makes one block cipher computation per *bit* of the input. However, that this may be too inefficient for, say, text files as input. Moreover, as for our previous schemes our construction also achieves ciphertext-integrity, whereas it seems hard to somehow modify the former to achieve such a notion.[2]

Security. The stronger security definition for an online cipher in [8] requires it to be indistinguishable from an "ideal" object that is a function drawn at random from a family of all possible such "online" permutations with the corresponding domain, even when given access to the corresponding "inverter" decryption oracle. Note that for example applying encryption character-by-character is completely insecure: encryptions of "HAT" and "BAT" should look totally unrelated in this setting despite sharing a suffix. We also formulate an additional property of ciphertext-integrity, and thus the encryption algorithm should contain some redundancy at the end so the ciphertext is verifiable. For our definition, we use an ideal object that encrypts a message with a random block appended, and the decryption oracle in the ideal experiment always returns $\perp$ to capture the intuition that the adversary should not be able to create a new valid ciphertext. The novelty of our definition is its generality: it uses only the ideal object in question and without any specific redundancy.

Definition 11. [Security of prefix-preserving ESAE] *Let $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a length- and prefix-preserving symmetric encryption scheme whose message*

[2] Of course, one can always achieve authenticity using a MAC on top of the encryption scheme, but the point is that this would be excessive in some applications.

space $\mathsf{MsgSp}(\mathcal{SE})$ *contains messages of multiple of block-length* n *and let* d *be the maximum possible number of blocks (hereafter we denote the set of such strings by* $D_{d,n}$*). Let* $\mathsf{OPerm}_{d,n}$ *denote the family of all length- and prefix-preserving permutations on* $D_{d,n}$*. Let* $\perp(\cdot)$ *denote the oracle that always returns* $\perp$ *and* r *denote a random* n*-bit block (picked fresh each time it is encountered). For an adversary* A *with access to two oracles define the experiments:*

Experiment $\mathbf{Exp}^{\text{pp-0}}_{\mathcal{SE},A}$	***Experiment*** $\mathbf{Exp}^{\text{pp-1}}_{\mathcal{SE},A}$
$sk \xleftarrow{\$} \mathcal{K}$; $d \xleftarrow{\$} A^{\mathcal{E}(sk,\cdot),\mathcal{D}(sk,\cdot)}$	$g \xleftarrow{\$} \mathsf{OPerm}_{d+1,n}$; $d \xleftarrow{\$} A^{g(\cdot\|\|r),\perp(\cdot)}$
Return d	*Return* d

We call A *a* pp-adversary *if it never repeats queries, never queries a response from its first oracle to its second, and all queries to its first oracle belong to* $D_{d,n}$ *and queries to its second belong to* $D_{d+1,n}$*. The advantage of a* A *is defined as*

$$\mathbf{Adv}^{\text{pp}}_{\mathcal{SE},A} = \Pr[\,\mathbf{Exp}^{\text{pp-0}}_{\mathcal{SE},A} = 0\,] - \Pr[\,\mathbf{Exp}^{\text{pp-1}}_{\mathcal{SE},A} = 0\,].$$

The scheme $\mathcal{SE}$ *is said to be* pp-secure *if for every efficient pp-adversary* A *the probability* $\mathbf{Adv}^{\text{pp}}_{\mathcal{SE},B}$ *is sufficiently small.* ▮

Discussion. Analogous to the case of exact-match queries, our security definition here ensures that the server cannot learn anything about the data except which attribute values share a same prefix, which is obviously unavoidable in this context, where the granularity of such prefix-correlation is given by the length of the block cipher used in our construction below (and on the other hand it is bit-wise for the less-efficient, no-authenticity scheme of [42]). Here one has to be wary of frequency-based (in terms how many *distinct* plaintexts with a given prefix occur in the database) deduction of some prefixes when using text data, which may require adding bogus data to balance these frequencies. We stress that this analysis holds *only* in the presence of prefix-matching (or exact-match) queries. In a generalization and refinement of the approach of [31] that we present in [4], we show that our scheme can in some sense be used to efficiently support range-queries as well, but the security analysis is more delicate.

Our Construction and Analysis. As in [8], appealing constructions such as the authenticated encryption scheme OCB [38] with fixed IV can be shown insecure under Definition 11. We design a prefix-preserving ESAE scheme based on an interesting modification of the HPCBC cipher [8, Construction 8.1] that appends an all-zero block to a message to encrypt and uses a different block cipher on this last block to also achieve ciphertext-integrity, which may also be of independent interest.[3] It is efficient and uses one block cipher and one hash function operation per block of input.

[3] In fact our construction treats HPCPC as a black-box so any on-line cipher that is OPRP-CCA (see [8] for the definition) can be used, but we suggest HPCBC for concreteness.

Definition 12. **[HCBC+]** *Let E: $\{0,1\}^{ek} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. Let H: $\{0,1\}^{hk} \times \{0,1\}^{2n} \to \{0,1\}^n$ be a family of functions. We associate to them a prefix-preserving ESAE scheme $\mathsf{HPCBC}^+ = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ defined as follows. The key generation algorithm chooses randomly a key $eK\|eK'\|hK$ where eK, eK' are (independent) keys for E and hK is a key for H. The encryption and decryption algorithms are defined as follows:*

```
Algorithm $\mathcal{E}(eK\|eK'\|hK, m)$
{ Parse $m$ as $m[1]\ldots m[l]$
$C[0] \leftarrow 0^n$ ; $m[0] \leftarrow 0^n$
For $i = 1, \ldots, l$ do
    $R \leftarrow m[i-1]||C[i-1]$
    $P[i] \leftarrow H(hK, R) \oplus m[i]$
    $C[i] \leftarrow E(eK, P[i]) \oplus H(hK, R)$}
$R \leftarrow m[l]||C[l]$
$P[l+1] \leftarrow H(hK, R) \oplus 0^n$
$C[l+1] \leftarrow E(eK', P[l+1]) \oplus H(hK,R)$
Return $C[1]\ldots C[l+1]$
```

```
Algorithm $\mathcal{D}(eK\|eK'\|hK, C)$
{Parse $C$ as $C[1]\ldots C[l+1]$ with $l \geq 1$
$C[0] \leftarrow 0^n$ ; $m[0] \leftarrow 0^n$
For $i = 1, \ldots, l$ do
    $R \leftarrow m[i-1]||C[i-1]$
    $P[i] \leftarrow E^{-1}(eK, C[i] \oplus H(hK, R))$
    $m[i] \leftarrow H(hK, R) \oplus P[i]$}
$R \leftarrow m[l]||C[l]$
$P[l+1] \leftarrow E^{-1}(eK', C[l+1] \oplus H(hK, R))$
$m[l+1] \leftarrow H(hK, R) \oplus P[l+1]$
If $m[l+1] = 0^n$ then return $m[1]\ldots m[l+1]$
Else return $\perp$
```

We note that the 6 first lines of the algorithms (i.e. the part between braces) could be expressed more compactly as $C[1]\ldots C[l] \leftarrow \mathsf{HPCBC}(eK\|hK, m)$ and $m[1]\ldots m[l] \leftarrow \mathsf{HPCBC}^{-1}(eK\|hK, C)$. This explicit description of HPCBC is given here for completeness. To see the benefit of using our construction over plain HPCBC note that encryption along with a separate MAC (e.g. CMAC) to additionally achieve integrity would roughly double the computation time, making two passes over the input, as compared to our construction.

Security of the scheme is based on the security of the underlying block cipher and the hash function. The corresponding definitions of PRP-CCA security of a block cipher and of almost-xor-universal hash functions is recalled in [8]. AES is believed to be PRP-CCA, and [8] provide references for secure hash function constructions. The proof of the following theorem is in [4]. It also contains the concrete security statement.

Theorem 3. *Let E: $\{0,1\}^{ek} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher that is a PRP-CCA. and let H: $\{0,1\}^{hk} \times \{0,1\}^{2n} \to \{0,1\}^n$ be an almost-xor-universal family of hash functions. Then HPCBC^+ defined via Definition 12 is a pp-secure prefix-preserving ESAE scheme.*

6.1 On Efficient Range-Query Processing

In [31] it is shown that encrypting data via a bit-wise prefix-preserving scheme allows efficient (as opposed to scanning the whole database) range queries over the data by specifying the possible prefixes for a desired range. Introducing our prefix-preserving ESAE as well provides a generalized approach, where the block size is not just one bit but a variable parameter. It is shown in [31] that certain attacks are possible if their scheme is used for range queries. In the full version of the paper [4], we generalize such attacks and discuss what is the best level of security prefix-preserving schemes can provide in this context.

Acknowledgments

We thank Brian Cooper and Andrey Balmin for useful comments and references.

References

1. The final HIPAA security rule. Federal Register (2003) Available at http://www.hipaadvisory.com/regs/finalsecurity/index.htm,
2. Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: A distributed architecture for secure database services. In: CIDR 2005
3. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: SIGMOD 2004
4. Amanatidis, G., Boldyreva, A., O'Neill, A.: New security models and provably-secure schemes for basic query support in outsourced databases. A full version of this paper (2007) Available at www-static.cc.gatech.edu/~aboldyre/publications.html
5. An, J.-H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, Springer, Heidelberg (2002)
6. Bellare, M.: Practice-oriented provable-security. In: Information Security Workshop, ISW (1997)
7. Bellare, M.: New proofs for NMAC and HMAC: Security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, Springer, Heidelberg (2006)
8. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: Online ciphers and the Hash-CBC construction. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, Springer, Heidelberg (2001)
9. Bellare, M., Boldyreva, A., O'Neill, A.: Efficiently-searchable and deterministic asymmetric encryption. Cryptology ePrint Archive, Report, /186, 2006. (2006), http://eprint.iacr.org/2006/186/
10. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, Springer, Heidelberg (1996)
11. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: FOCS (1997)
12. Bellare, M., Kohno, T., Namprempre, C.: Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm. In: ACM Transactions on Information and System Security. vol. 7(2) (2004)
13. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, Springer, Heidelberg (2000)
14. Black, J., Rogaway, P.: CBC MACs for arbitrary-length messages: The three-key constructions. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, Springer, Heidelberg (2000)
15. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, Springer, Heidelberg (2004)

16. Canetti, R., Krawczyk, H., Nielsen, J.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, Springer, Heidelberg (2003)
17. Ceselli, A., Damiani, E., De Capitani, d.S., Jajodia, S., Paraboschi, S., Samarati, P.: Modeling and assessing inference exposure in encrypted databases. ACM Trans. Inf. Syst. Secur. 8(1), 119–152 (2005)
18. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, Springer, Heidelberg (2005)
19. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions. Cryptology ePrint Archive, Report 2006/210 (2006)
20. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: Computing range queries on obfuscated data. In: Information Processing and Management of Uncertainty in Knowledge-Based Systems (2004)
21. Damiani, E., De Capitani Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational DBMSs. In: CCS (2003)
22. Goh, E.-J.: Secure indexes. Cryptology ePrint Archive, Report 2003/216 (2003) `http://eprint.iacr.org/2003/216/`.
23. Goldwasser, S., Micali, S.: Probabilistic encryption. In: Journal of Computer and Systems Sciencies, vol. 28 (1984)
24. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Applied Cryptography and Network Security Conference
25. Hacigümüs, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: SIGMOD (2002)
26. Hacigümüs, H., Iyer, B.R., Mehrotra, S.: Efficient execution of aggregation queries over encrypted relational databases. In: Lee, Y., Li, J., Whang, K.-Y., Lee, D. (eds.) DASFAA 2004. LNCS, vol. 2973, Springer, Heidelberg (2004)
27. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: VLDB (2004)
28. Iyer, B.R., Mehrotra, S., Mykletun, E., Tsudik, G., Wu, Y.: A framework for efficient storage security in RDBMS. In: EDBT (2004)
29. Kantracioglu, M., Clifton, C.: Security issues in querying encrypted data. In: DBSec (2005)
30. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: SIGMOD, ACM Press, New York (2006)
31. Li, J., Omiecinski, E.: Efficiency and security trade-off in supporting range queries on encrypted databases. In: DBSec (2005)
32. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. In: NDSS (2004)
33. Mykletun, E., Tsudik, G.: Incorporating a secure coprocessor in the database-as-a-service model. In: International Workshop on Innovative Architecture for Future Generation High Performance Processors and Systems (2005)
34. Mykletun, E., Tsudik, G.: Aggregation queries in the database-as-a-service model. In: DBSEC (2006)
35. Narasimha, M., Tsudik, G.: DSAC: integrity for outsourced databases with signature aggregation and chaining. In: CIKM (2005)
36. Narasimha, M., Tsudik, G.: Authentication of outsourced databases using signature aggregation and chaining. In: Lee, M.L., Tan, K.-L., Wuwongse, V. (eds.) DASFAA 2006. LNCS, vol. 3882, Springer, Heidelberg (2006)

37. Özsoyoglu, G., Singer, D.A., Chung, S.S.: Anti-tamper databases: Querying encrypted databases. In: DBSec, pp. 133–146 (2003)
38. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: ACM CCS (2001)
39. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, Springer, Heidelberg (2006)
40. Arsenal Digital Solutions. Top 10 reasons to outsource remote data protection. http://www.arsenaldigital.com/services/remote_data_protection.htm
41. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: IEEE Symposium on Security and Privacy (2000)
42. Xu, J., Fan, J., Ammar, M.H., Moon, S.B.: Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In: ICNP (2002)

Authenticated Relational Tables and Authenticated Skip Lists*

Giuseppe Di Battista[1] and Bernardo Palazzi[1,2,3]

[1] Roma TRE University, Rome Italy
{gdb,palazzi}@dia.uniroma3.it,
[2] ISCOM Italian Ministry of Communication, Rome, Italy
[3] Brown University, Department of Computer Science, Providence, RI USA

Abstract. We present a general method, based on the usage of typical DBMS primitives, for maintaining authenticated relational tables. The authentication process is managed by an application external to the DBMS, that stores just one hash information of the authentication structure. The method exploits techniques to represent hierarchical data structures into relational tables and queries that allow an efficient selection of the elements needed for authentication.

Keyword: Authenticated Relational Table, Authenticated Skip List, Authenticated query.

1 Introduction

We consider the following scenario. A user needs to store data in a relational database, where the Data Base Management System (DBMS) is shared with other users. For example, the DBMS is available on-line through the Web, and anybody in the Internet can store and access data on it. Nowadays, there are many sites providing services of this type [1,22,25,30] and the literature refers to such facilities as to *outsourced databases* [13,19,27].

When the database is accessed, the user wants to be sure on the integrity of her/his data, and wants to have the proof that nobody altered them.

Of course, accessing the DBMS is subject to authentication restrictions, and the users must provide credentials to enter. However, the user might not trust the DBMS manager, or the site that provides the service, or even the DBMS software. Extending the argument, the same problem can be formulated even in terms of a traditional database. Also in this case, with the current technologies, although DBMSes put at disposal logs of the performed transactions and other security features, for a user it is somehow impossible to be completely sure that nobody altered the data.

A first attempt for the user to be sure of the authenticity of the data is to put a signature on each t-uple of each relational table of the database. Unfortunately, this technique does not provide enough security. In fact, adversaries could

* This work was supported in part by grant IIS–0324846 from the National Science Foundation and a gift from IAM Technology, Inc.

remove some t-uples and the user would not have any evidence of this. Another straightforward possibility would be to sign each table as a whole. However, this does not scale-up, and even mid-size tables would be impossible to authenticate.

We propose a method and a prototype for solving the above mentioned problem. Namely, for each relational table R of the user we propose to store in an extra relational table $S(R)$ (in the following *security table*) of the DBMS a special version of authenticated data structure that allows to verify the authenticity of R.

With this approach, if the user wants to have the proof of authenticity of R, it is sufficient to check the values of a few elements stored in $S(R)$. On the other hand, if the user updates R, only a few variations on $S(R)$ are needed to preserve the proof of authenticity. We also propose efficient techniques to manage and to query $S(R)$ and show the practical feasibility of the approach.

Observe that the proposed approach is completely independent on the specific adopted DBMS and can be implemented into an extra software layer or either a plug-in, under the sole responsibility of the user. The authentication process is managed by an application external to the DBMS that stores just a constant size ($O(1)$ wrt the size of R) secret. The method does not require trust in the DB manager or DBMS.

The paper is organized as follows. Section 2 provides basic terminology and summarizes the state of the art. Section 3 describes the adopted model. Sections 4 and 5 provide technical insights. Section 6 presents experiments that show the feasibility of the approach. Section 7 concludes the paper analyzing the security of the approach and proposing future work.

2 Background and State of the Art

Authenticated data structures (ADS) [29] have been devised to be used in a computational model where untrusted responders answer queries on a data structure on behalf of a trusted source and provide to the user a proof of the validity of the answer. Early work on ADS was originated by the certificate revocation problem in $PKIs$ and focused the attention on authenticated dictionaries, on which membership queries are performed.

The Merkle hash tree MHT scheme [16] can be used to implement a static authenticated dictionary. An MHT of a set stores cryptographic hashes of the value of elements belonging to the set at the leaves of the MHT and an authentication value at each internal node, which is the result of computing a cryptographic hash function on the values of its children. The MHT uses linear space and has $O(\log n)$ proof size, query time and verification time.

A dynamic authenticated dictionary that uses a hierarchical hashing technique over skip lists, a data structure introduced by Pugh [26], is presented in [8,9]. Such ADS obtains $O(\log n)$ proof size, query time, update time and verification time. Other schemes based on variations of MHT have been proposed in [2,4,12,20]. A detailed analysis of the efficiency of authenticated dictionary schemes based on hierarchical cryptographic hashing is conducted in [28], where precise measures of the computational overhead due to the authentication are introduced. Lower bounds on the authentication cost are given, existing authentication schemes are analyzed, and a new authentication scheme is presented that achieve performance very close to the theoretical optimal.

The notion of a two parties model in ADS is introduced in [10], where only the client needs to maintain the proof of validity for his data.

A first step towards the design of more general ADS (beyond dictionaries) is done in [7,14,21] with a first approach on the authentication of relational database operations and multidimensional orthogonal range queries.

Buldas, in a more recent paper [3], studies how to extend ADS to perform more complex queries and uses optimizations on interval queries. In [23,24] the authors propose a method to authenticate projection queries using different cryptographic techniques for verifying the completeness of relational queries. While the papers are quite promising in terms of theoretical bounds and analysis, the practical efficiency is not demonstrated.

In [17], Miklau and Suciu proposed to embed into a relational table an MHT, with a model that is similar to the one adopted in this paper. However, the technique is described only partially and seems to have some drawbacks. Namely, validating the result of a query seems to require several distinct queries on the DBMS. This is in contrast with the typical atomicity requirements of concurrency. Also, the $MHTs$ require frequent rebalancing for supporting updates and it is unclear how to match this requirement with the need to have a few updates in the relational table. Further, the time performance illustrated in the paper are not supported by a clear description of the experimental platform and show some inconsistency. For example in one of the tests the time requested for authentication decreases with the growth of the table.

For the purposes of this paper we need to provide a description of the skip lists. The skip list data structure [26] is an efficient tool for storing an ordered set of *elements*. It supports the following operations on a set of elements.

– **find**(x): Determine whether element x is in the set.
– **insert**(x): Insert element x into the set.
– **delete**(x): Remove element x from the set.

A skip list S stores a set of elements in a sequence of linked lists $S_0, S_1, \ldots, S_t$ called *levels*. The members of the lists are called *nodes*. The base list, S_0, stores in its nodes all the elements of S in order, as well as *sentinels* associated with the special elements $-\infty$ and $+\infty$. Each list S_{i+1} stores a subset of the elements of S_i. The method used to define the subset from one level to the next determines the type of skip list. The default method is simply to choose the elements of S_{i+1} at random among the elements of S_i with probability $\frac{1}{2}$. One could also define a deterministic skip list [18], which uses simple rules to guarantee that between any two elements in S_i there are at least 1 and at most 3 elements of S_{i+1}. In either case, the sentinel elements $-\infty$ and $+\infty$ are always included in the next level up, and the top level, is maintained to be $O(\log n)$. We therefore distinguish the node of the top list S_t storing $-\infty$ as the start node s.

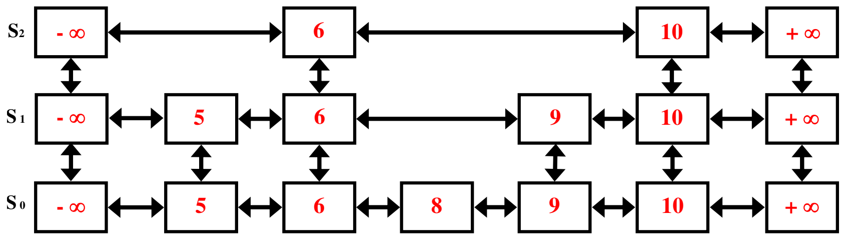


Fig. 1. Skip List

An element that is in S_{i-1} but not in S_i is said to be a *plateau element* of S_{i-1}. An element that is in both S_{i-1} and S_i is said to be a *tower element* in S_{i-1}. Thus, between any two tower elements, there are some plateau elements. In randomized skip lists, the expected number of plateau elements between two tower elements is one. The skip list of Fig. 1 has 7 elements (including sentinels). The element 6 is stored in 3 nodes with different level. The overall number of nodes is 17.

To perform a search for element x in a skip list, we begin at the start node s. Let v denote the current node in our search (initially, $v = s$). The search proceeds using two actions, *hop forward* and *drop down*, which are repeated one after the other until we terminate the search. See Fig. 2.

- $Hopforward$: We move right along the current list until we find the node of the current list with largest element less than or equal to x. That is, while $elem(right(v)) < x$, we perform $v = right(v)$.
- *Dropdown*: If $down(v) = null$, then we are done with our search: node v stores the largest element in the skip list less than or equal to x. Otherwise, we update $v = down(v)$.

In a deterministic skip list, the above searching process is guaranteed to take $O(\log n)$ time. Even in a randomized skip list, it is fairly straightforward to show (e.g., see [11]) that the above searching process runs in expected $O(\log n)$ time, for, with high probability, the height t of the randomized skip list is $O(\log n)$ and the expected number of nodes visited on any level is 3.

To insert a new element x, we determine which lists should contain the new element x by a sequence of simulated random coin flips. Starting with $i = 0$, while the coin comes up heads, we use the stack A to trace our way back to the position of list S_{i+1} where element x should go, add a new node storing x to this list, and set $i = i + 1$. We continue this insertion process until the coin comes up tails. If we reach the top level with this insertion process, we add a new top level on top of the current one. The time taken by the above insertion method is $O(\log n)$ with high probability. To delete an existing element x, we remove all the nodes that contain the element x. This takes time is $O(\log n)$ with high probability.

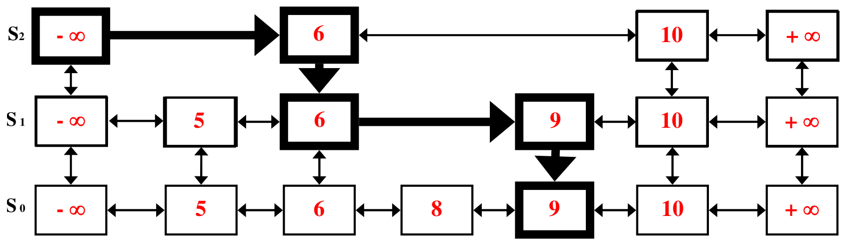


Fig. 2. A value searching in a Skip List: search for element 9 in the skip list of Figure 1. The nodes visited and the links traversed are drawn with thick lines and arrows.

To introduce the *Authenticated Skip Lists* we need to use the commutative hash technique [9] developed by Gooodrich and Tamassia. A hash function h is commutative if $h(x; y) = h(y; x)$, for all x and y. Given a cryptographic hash function h that is collision resistant in the usual sense, we construct a candidate commutative cryptographic hash function, h_0, as follows [9] :

$h_0(x, y) = h(min(x, y), max(x, y))$

It can be shown that h_0 is commutatively collision resistant [9].

The authenticated skip list introduced in [9] consists of a skip list where each node v stores a label computed accumulating the elements of the set with a commutatively cryptographic hash function h. For completeness, let us review how hashing occurs. See [9] for details. For each node v we define label $f(v)$ in terms of the respective values at nodes $w = right(v)$ and $u = down(v)$. If $right(v) = null$, then we define $f(v) = 0$. The definition of $f(v)$ in the general case depends on whether u exists or not for this node v.

- $u = null$, i.e., v is on the base level:
  - If w is a tower node, then $f(v) = h(elem(v), elem(w))$
  - If w is a plateau node, then $f(v) = h(elem(v), f(w))$.
- $u \neq null$, i.e., v is not on the base level:
  - If w is a tower node, then $f(v) = f(u)$.
  - If w is a plateau node, then $f(v) = h(f(u), f(w))$.

We illustrate the flow of the computation of the hash values labeling the nodes of a skip list in See Fig. 3. Note that the computation flow defines a directed acyclic graph DAG, not a tree. After performing the update in the skip list, the hash values must be updated to reflect the change that has occurred. The additional computational expense needed to update all these values is expected with high probability to be $O(\log n)$. The verification of the answer to a query is simple, thanks to the use of a commutative hash function. Recall that the goal is to produce a verification that some element x is or is not contained in

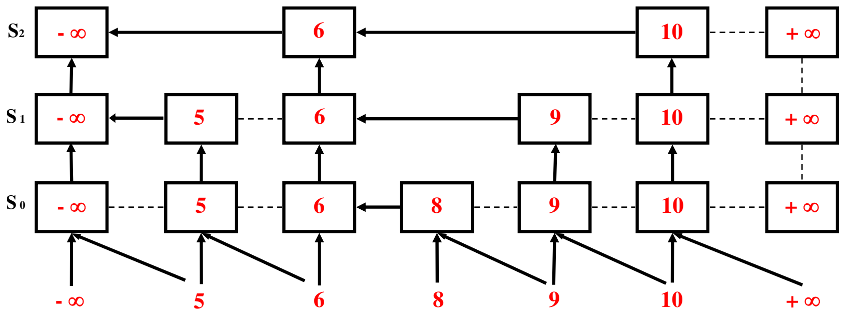


Fig. 3. Authenticated Skip List: Flow of the computation of the hash values labeling the nodes of the skip list of Fig. 2. Nodes where hash functions are computed are drawn with thick lines. The arrows denote the flow of information, not links in the data structure.

the skip list. In the case when the answer is "*yes*", we verify the presence of the element itself. Otherwise, we verify the presence of two elements x_a and x_b stored at consecutive nodes on the bottom level S_0 such that $x_a < x < x_b$. In either case, the answer authentication information is a single sequence of values, together with the signed, timestamped, label $f(s)$ of the start node s.

Let $P(x) = (v_1; ...; v_m)$ be the sequence of nodes that are visited when searching for element x, in reverse order. In the example of Fig. 4, we have $P(9)$ that needs not only the nodes $(9, 6, -\infty)$ with the thick line but also all the siblings with the stroke dash-dot-dash-dot. Note that by the properties of a skip list, the size m of sequence $P(x)$ is $O(\log n)$ with high probability. We construct from the node sequence $P(x)$ a sequence $Q(x) = (y_1; ...; y_m)$ of values such that:

- $y_m = f(s)$, the label of the start node;
- $y_m = h(y_{m-1}; h(y_{m-2}; h(...; y_1)...)))$

The user verifies the answer for element x by simply hashing the values of the sequence $P(x)$ in the given order, and comparing the result with the signed value $f(s)$, where s is the start node of the skip list. If the two values agree, then the user is assured of the validity of the answer at the time given by the timestamp.

3 The Reference Model

A user stores a relational table R into a DBMS. The user would like to perform the usual relational operations on R, namely, would like to select a set of t-uples, to insert elements, and to delete elements. The user wants to verify that a query result is authentic. The amount of information that the user has to maintain in a secure environment to be certain of the authenticity of the answer should be kept small (ideally constant size) with respect to the size of R.

We propose to equip R with an authenticated skip list A to guarantee its integrity. Of course, there are at least two approaches for implementing A. Either

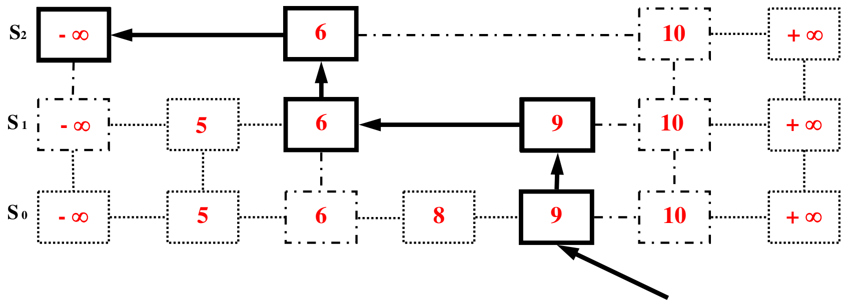


Fig. 4. Values needed to authenticate the result of a query

A is stored in main memory within an application controlled by the user, or A is stored into the same DBMS storing R. We follow the second approach. Namely, we investigate how to efficiently store A into a further relational table $S(R)$, called *security table*, used only for that purpose. Fig. 5 shows a relational table, an authenticated skip list for its elements, and the implementation of the skip-list into a second relational table.

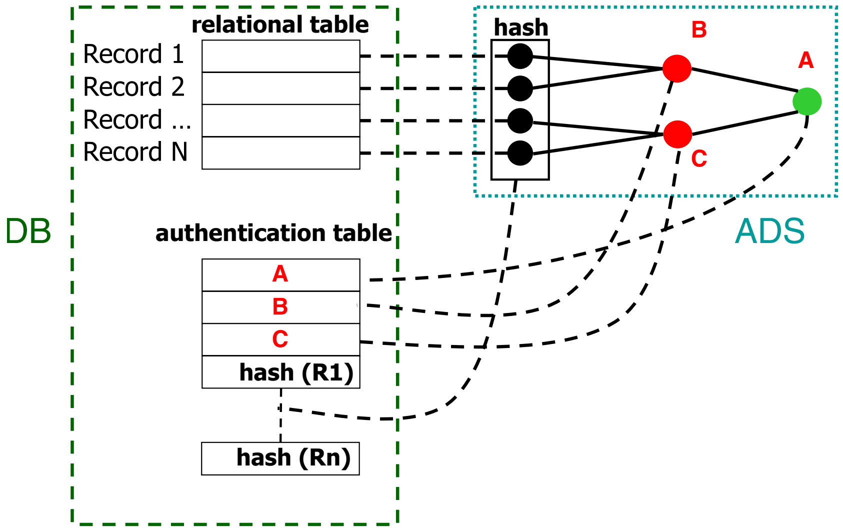


Fig. 5. A relational table and its security table

There are two options. We call them the *coarse-grained* and the *fine-grained* approach.

What we call coarse-grained approach is probably the most natural way to represent an authenticated skip list S inside a relational table $S(R)$. Namely, it consists of storing each element of S inside a specific record of $S(R)$. On the other hand, the fine-grained approach shifts the attention on a smaller element of S. It consists of storing each level of an element of S inside a record of $S(R)$.

In order to visualize the coarse-grained approach, it is effective to think at S in terms of a "quarter clockwise rotation". As an example, Table 1 is a coarse-grained representation of the authenticated skip list of Fig. 6.

More precisely, the fields of Table 1 have the following meaning.

– **Key**: The value of an element of S. It can be any type of value, not only a number, but on such a type a total ordered must be defined.
– **Prv** n - **Nxt** n: Pointers to the previous and to the next element in S, for each level n.
– **Hash** n: Information needed to authenticate S, stored at each level n.

Each element of S has a height, that is, the number of nodes with the same value of key that constitute an element of S, that is randomly determined. This is the main trade-off of this technique, because on one hand this kind of representation has the property to maintain the identity between the number of records in $S(R)$ and the elements present in S, but on the other hand it has an overhead in the size of the table, because each record has a number of fields equal to the highest S in A. This is necessary because we do not know the height of a new S and then we have to arrange $S(R)$ for worst cases, when an S is at the highest level. So, we must pad with "null" values the fields that do not reach the highest level.

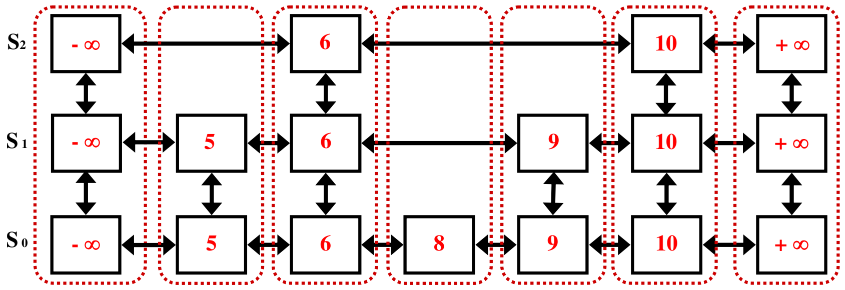


Fig. 6. Storing a Skip List inside a Relational Table

Once stated how to represent S inside the security table $S(R)$, we developed methods to perform in S a set of authenticated relational operations, without the need to load in main memory the whole $S(R)$. Performing authenticated operations on R requires the usage of queries that retrieve all the elements that are needed to compute the authentication path. Such elements are spread on all $S(R)$. The main requirements in devising such queries are:

– The need to build queries that retrieve only the authentication elements that are strictly necessary, to reduce, as much as possible, the amount of required memory.

Table 1. A coarse-grain representation of an authenticated skip list into a relational table. In bold face the elements necessary to authenticate element 9.

Key	**Hash 0**	**Prv 0**	**Nxt 0**	**Hash 1**	**Prv 1**	**Nxt 1**	**Hash 2**	**Prv 2**	**Nxt 2**
- ∞	f($-\infty$, 5)	*null*	5	**f(f($-\infty$), f(5))**	*null*	5	**f(f($-\infty$),f(6))**	*null*	**6**
5	f(5, 6)	$-\infty$	6	f(f(5), f(6))	$-\infty$	6	*null*	*null*	*null*
6	**f(6, f(8))**	5	8	**f(f(6), f(9))**	5	**9**	$f(f(6), f(10))$	10	$-\infty$
8	f(9,6)	9	6	*null*	*null*	*null*	*null*	*null*	*null*
9	**f(9, 10)**	8	10	**f(9, 10)**	**6**	**10**	*null*	*null*	*null*
10	f(10, f($+\infty$))	9	$+\infty$	**f(f(10), f($+\infty$))**	9	$+\infty$	**f(f(10), f($+\infty$))**	6	$+\infty$

– The need of fast queries that allow to authenticate a result with a small time overhead. In this respect it is meaningful to minimize the number of used queries.

It is important to perform such queries using only standard SQL. In fact, our model does not allow any modification of the DBMS engine. Also, thinking in terms of SQL allows the identification of a precise interface between an authentication tool based on our techniques and the DBMS, allowing its implementation in terms of a plug-in. The main idea here is to use an algorithm that retrieves the authentication elements, starting from the knowledge of the value K to authenticate:

1. We perform a query that loads in memory all the records that are not *null* at top level and that have a value smaller than K.
2. We select the greatest element in the query result (that is the predecessor of K at the top level).
3. We perform an interval query on the elements (that are not *null*) at the immediately lower level, with the following range: from the element retrieved in the previous step to the element stored in its field next to the top level.
4. We repeat the steps $2-3$ until we reach level 0.

In order to understand which elements are loaded in main memory by queries of the algorithm, it is effective to think at a shape like a "funnel" that has its stem on K. See Fig. 7. The loaded elements are those that "touch" the funnel.

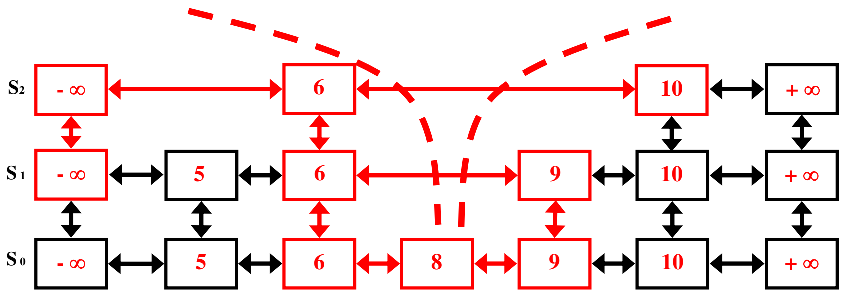


Fig. 7. Loaded elements in an authentication query

Note that the number of queries that is needed to retrieve the authentication root path is proportional to the number of levels in S, that is logarithmic in the number of elements that are currently present in S.

4 A Fine Grained Approach

This approach stores inside each record a node instead of an element of S. A node is an invariant-size component in S. Hence, it can be stored in a record with a fixed number of fields, independently on the number of elements stored in S. More precisely, in this case the fields of $S(R)$ have the following meaning:

- **Key**: value of an element of S;
- **Level**: height of an element of S, that is the number of the lists that the element belongs to;
- **prvKey-nxtKey**: pointers to the previous and to the next element of S at the same level;
- **parentLvl-parentKey**: pointer to the parent element in the path of authentication; it is needed to allow the retrieval of the root path;
- **Hash**: information needed for the authentication, performed with the method used in S [9].

The direct storage of S nodes significantly reduces the space overhead, that it is typical of the coarse grain approach. In fact, in this case there is no need to store *null* values.

This approach allows the usage of very efficient techniques to manage $S(R)$ dynamically and securely. The method we adopt is based on the *nested set* method for storing hierarchical data structures inside adjacency lists, that in turn fit well into relational tables [5] See Fig. 8 and Tab. 2.

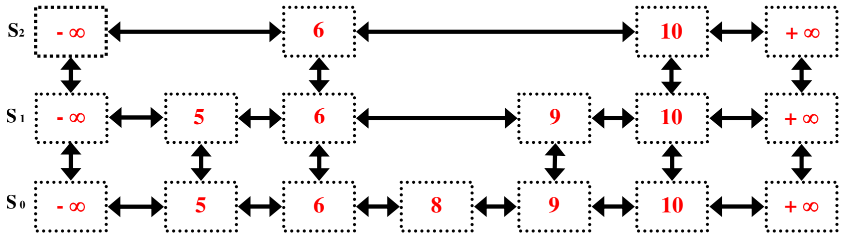


Fig. 8. Storing a Skip List inside a Relational Table. A Fine Grained Approach.

5 Exploiting Nested Sets

The problem of storing hierarchical data structures inside relational tables has been already studied in database theory [6,15]. The solution that we exploit is

Table 2. A fine grain representation of an authenticated skip list into a relational table. In bold the elements necessary to authenticate element 9.

Key	**Level**	**prvKey**	**nxtKey**	**parentLvl**	**parentKey**	**Hash**
$-\infty$	**2**	*null*	6	*null*	*null*	**f(f($-\infty$), f(6))**
$-\infty$	**1**	*null*	5	2	$-\infty$	**f(f($-\infty$), f(5))**
$-\infty$	**0**	*null*	5	1	$-\infty$	**f($-\infty$, 5)**
5	1	$-\infty$	6	1	$-\infty$	f(5,6)
5	0	$-\infty$	6	1	5	f(5, 6)
6	**2**	$-\infty$	**10**	**2**	$-\infty$	f(**f(6),f(10)**)
6	**1**	5	**9**	**2**	**6**	**f(f(6),f(9))**
6	**0**	5	8	1	6	**f(6, f(8))**
8	0	6	9	0	6	f(8, 10)
9	**1**	**6**	**10**	**1**	**6**	**f(9, 10)**
9	**0**	8	10	1	9	**f(9, 10)**
10	**2**	6	$+\infty$	2	6	**f(10,f($+\infty$))**
10	**1**	9	$+\infty$	2	10	**f(10,f($+\infty$))**
10	0	9	$+\infty$	2	10	f(10,f($+\infty$))

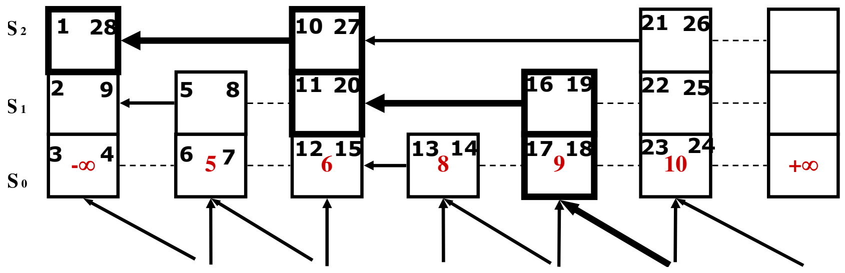


Fig. 9. An ADS and its Nested Set. Thick lines show the authentication root path for element 9.

due to Celko [5], that shows a method to store a tree inside a relational table. Such a method is based on augmenting the table with two extra fields.

In order to understand what is a nested set, it is effective to think at the nodes of the tree as circles and to imagine that the circles of the children are nested inside their parent. The root of the tree is the largest circle and contains all the other nodes. The leaf nodes are the innermost circles, with nothing else inside them. The nesting shows the hierarchical relationship.

The two extra fields have the role of left and right boundaries of the circle and allow to represent the nesting of the hierarchy.

Unfortunately, skip lists are not *trees* but a directed acyclic graph. Hence, we have to extend the nested set method to this different setting. Table 3 illustrates how the fine-grained approach can be equipped with nested-sets features. Observe the Left and Right fields that represent the boundaries of the "circles".

Table 3. A representation of an authenticated skip list into a relational table using nested set. In bold the key value and the *left* and *right* fields. The 2 extra fields added are needed for fast queries.

Key	**Level**	**prvKey**	**nxtKey**	**parentLvl**	**parentKey**	**Left**	**Right**
$-\infty$	**2**	*null*	6	*null*	*null*	**1**	**28**
$-\infty$	**1**	*null*	5	2	$-\infty$	**2**	**9**
$-\infty$	**0**	*null*	5	1	$-\infty$	**3**	**4**
5	**1**	$-\infty$	6	1	$-\infty$	**5**	**8**
5	**0**	$-\infty$	6	1	5	**6**	**7**
6	**2**	$-\infty$	10	2	$-\infty$	**10**	**27**
6	**1**	5	9	2	6	**11**	**20**
6	**0**	5	8	1	6	**12**	**15**
8	**0**	6	9	0	6	**13**	**14**
9	**1**	6	10	1	6	**16**	**19**
9	**0**	8	10	1	9	**17**	**18**
10	**2**	6	$+\infty$	2	6	**21**	**26**
10	**1**	9	$+\infty$	2	10	**22**	**25**
10	**0**	9	$+\infty$	2	10	**23**	**24**

Fig. 9 shows the correspondence between boundaries and nodes of the skip-list. The figure shows also a root path.

Now we show one of the features of the proposed approach. Namely, we argue that, in order to authenticate an element of a relational table R, we need just one query on $S(R)$. Such a query is used to retrieve the complete *root-path* and all its *sibling* elements. Observe that, authenticating an element in an ADS requires a number of steps that is logarithmic (worst case or average case) in the number of the elements while this logarithmic dependence does not yield a logarithmic number of queries in our case but a constant number of queries. We make the argument using an example. The following query uses directly the value of the element to authenticate. The example is for the authentication of element 9.

```
SELECT    *
FROM          skiplist
WHERE  Left <= (SELECT     Left
                FROM      skiplist
                WHERE      key =  9 AND level = 0)
       AND  Right >= (
                        SELECT     Right
                        FROM          skiplist
                        WHERE      key =  9 AND level = 0);
```

The above query retrieves only the authentication root-path starting from 9. To validate 9 we have to retrieve also all sibling nodes of the root-path. This is possible by using two subqueries that retrieve all elements that are:

- in the fields nxtKey of the root-path;
- on the level below and with the same key of the root-path.

Using this method we built a quick algorithm to get the complete authentication path needed to validate a table interrogation, using only one query, that is that all concurrency problems related to selection queries will be managed by the DBMS. Also, it is possible to modify the query in order to retrieve all the information needed to authenticate all the *t*-uples obtained by a Select with just one query.

6 Experimental Evaluation

This section shows the experimental results obtained using a prototype implementation of the techniques presented in the previous sections. The Hardware architecture where tests have been performed consists of quite common laptop with following features:

- cpu intel©centrino^{TM}duo T2300 (1.66 GHz, 667 FSB);
- RAM 1.5 Gb DDR2
- HDD 5,400 rpm Serial ATA

The Software architecture consists of following elements:

- Microsoft©Windows^{TM}XP Tablet edition 2005;
- Java^{TM}version 1.5
- MySql JDBC Connector Java-bean 5.03
- MySql DBMS version 4.1

The data sets for tests have been chosen with a scale from $10,000$ to $1,000,000$ of elements. Such elements were sampled at random from a set 10 times larger. All values presented in this section have been computed as average of the results of 5 different tests. The elements in each test are a sample, randomly selected, composed of $\frac{1}{1000}$ of the entire set. All times are in *milliseconds*. All tests show the *clock-wall* time.

The first test is about the authentication of a single value inside a relational table. Table 4 shows the results of the authentication of a single element inside different size authenticated tables, stressing the differences between coarse grain and fine grain approaches. Tests are about the following measures:

- **RAM:** the time to validate a value in main memory;
- **DB → RAM:** the time to load in main memory from a secondary memory storage system (e.g., a hard disk), the elements necessary to validation;
- **NODES:** the numbers of elements loaded from the database in main memory;
- **STEPS:** the numbers of elements actually used in the authentication process, the difference between NODES value and this value shows the overhead of the elements loaded in main memory.

Table 4. Test results for validation of an element inside different size tables. All the results are in *ms*. Times for fine- and coarse-grained approaches.

CHECK	10,000		100,000		1,000,000	
	Coarse	Fine	Coarse	Fine	Coarse	Fine
RAM	0	0	0	0	0	0
DB → RAM	36	11	252	42	2680	377
NODES	35	27	44	31	57	43
STEPS	25	27	33	30	39	41

Table 5. Test results for insertion of an element inside a different size tables. Using coarse grain approach. All results are in *ms*.

INSERT	10,000	100,000	1,000,000
RAM	0	0	0
DB → RAM	32	260	2605
RAM → DB	14	26	26
Tot. Time	46	286	2631

The results showed above are very similar to those obtained from the authentication of an element not-present in the table. In fact it is sufficient to check the previous and the next element of the value that is not present to proof the element lack.

The second test is about the insertion of a single value inside an authenticated relational table. The table 5 shows the results of the insertion of a single element inside different size authenticated tables using only coarse grain approach. Tests concern the following measures:

- **RAM**: the time to insert in main memory a value;
- **DB → RAM**: the time to load in main memory from a secondary memory storage system (e.g., a hard disk), the elements necessary to insertion;
- **RAM → DB**: the time to store in secondary memory the elements updated in main memory;

Methods that allow to delete and modify an element inside an authenticated table are similar to times showed for insertion operation.

The obtained experimental results put in evidence the feasibility of the approach. In fact, the time for answering a query is comparable to the one obtained in a non authenticated setting. The fine-grained approach, based on Celko techniques, shows much better performance wrt the coarse-grained one.

7 Conclusions and Future Work

We have described methods that allow a user to verify the authenticity and completeness of simple queries results, even if the database system is not trusted.

The overhead for the user is limited at storing only a single hash value. Our work is the first to design and evaluate techniques for authenticated skip list that are appropriate to a relational database, and the first to prove the feasibility of authenticated skip list for integrity of databases.

The security of the presented method is based on the reliability of *ADSes*. There are many works [3,9,12,16] in the literature that demonstrate that the security of *ADS* is based on the difficulty to find useful collisions in a cryptographic hash function. So all the security relies on the effectiveness of hash functions. The prototype used for the experiments uses commutative hashing. In [9] it is demonstrated that commutative hashing does not augment the possibility to find a collision in the used hash function.

In the future we would like to investigate how to authenticate more complex queries making use of a larger set of relational operations. Further, we would like to study models to build integrity verification services in peer to peer systems.

References

1. Web based Database Software Solutions On-Demand. `http://www.teamdesk.net`
2. Buldas, A., Laud, P., Lipmaa, H.: Accountable certificate management using undeniable attestations. In: ACM Conference on Computer and Communications Security, pp. 9–17 (2000)
3. Buldas, A., Roos, M., Willemson, J.: Undeniable replies for database queries. In: Proceedings of the Fifth International Baltic Conference on DB and IS, 2002. vol. 2, pp. 215–226 (2002)
4. Carminati, B.:Selective and authentic third-party distribution of xml documents. IEEE Transactions on Knowledge and Data Engineering. Fellow-Elisa Bertino and Member-Elena Ferrari and Fellow-Bhavani Thuraisingham and Senior Member-Amar Gupta. vol.16(10), pp.1263–1278 (2004)
5. Celko, J.: Joe Celko's Trees and hierarchiesin SQL for smarties. Morgan-Kaufmann, Seattle, Washington, USA (2004)
6. Date, C.J.: Why is it so difficult to provide a relational interface to ims. In: Relational Database– Selected Writings, pp. 241–257. Addison-Wesley, London (1986)
7. Devanbu, P.T., Gertz, M., Martel, C.U., Stubblebine, S.G.: Authentic third-party data publication. In: Proceedings of the IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security, Deventer, The Netherlands, pp. 101–112. Kluwer Academic Publishers, Dordrecht (2001)
8. Goodrich, M., Schwerin, A., Tamassia, R.: An efficient dynamic and distributed cryptographic accumulator. Technical report, Johns Hopkins Information (2000)
9. Goodrich, M., Tamassia, R.: Efficient authenticated dictionaries with skip lists and commutative hashing. Technical report, Johns Hopkins Information (2000)
10. Goodrich, M.T., Shin, M., Tamassia, R., Winsborough, W.H.: Authenticated dictionaries for fresh attribute credentials. In: Nixon, P., Terzis, S. (eds.) iTrust 2003. LNCS, vol. 2692, Springer, Heidelberg (2003)
11. Goodrich, M.T., Tamassia, R.: Data Structures and Algorithms in Java. John Wiley & Sons, Inc, New York, NY, USA (2000)
12. Kocher, P.C.: On certificate revocation and validation. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 172–177. Springer, Heidelberg (1998)

13. Li, F., Hadjieleftheriou, M., Kollios, G., Reyzin, L.: Dynamic authenticated index structures for outsourced databases. In: SIGMOD '06. Proceedings of the 2006 ACM SIGMOD international conference on Management of data, New York, NY, USA, pp. 121–132. ACM Press, New York (2006)
14. Martel, C., Nuckolls, G., Devanbu, P., Gertz, M., Kwong, A., Stubblebine, S.G.: A general model for authenticated data structures. Algorithmica 39(1), 21–41 (2004)
15. Meier, A., Dippold, R., Mercerat, J., Muriset, A., Untersinger, J., Eckerlin, R., Ferrara, F.: Hierarchical to relational database migration. IEEE Softw. 11(3), 21–27 (1994)
16. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
17. Miklau, G., Suciu, D.: Implementing a tamper-evident database system. In: ASIAN: 10th Asian Computing Science Conference, pp. 28–48 (2005)
18. Ian Munro, J., Papadakis, T., Sedgewick, R.: Deterministic skip lists. In: SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, pp. 367–375 (1992)
19. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced databases. Trans. Storage 2(2), 107–138 (2006)
20. Naor, M., Nissim, K.: Certificate revocation and certificate update. In: Proceedings 7th USENIX Security Symposium (January 1998)
21. Nuckolls, G., Martel, C., Stubblebine, S.: Certifying data from multiple sources. In: EC '03. Proceedings of the 4th ACM conference on Electronic commerce, New York, NY, USA, pp. 210–211. ACM Press, New York (2003)
22. Caspio Bridge online database. http://www.caspio.com
23. Pang, H., Jain, A., Ramamritham, K., Tan, K.: Verifying completeness of relational query results in data publishing. In: SIGMOD Conference, pp. 407–418 (2005)
24. Pang, H., Tan, K.: Authenticating query results in edge computing. In: ICDE '04. Proceedings of the 20th International Conference on Data Engineering, Washington, DC, USA, p. 560. IEEE Computer Society, Los Alamitos (2004)
25. Livebase project Blog on Web-based db. http://livebase.blog.com/1142527/
26. Pugh, W.: Skip lists: A probabilistic alternative to balanced trees. In: Workshop on Algorithms and Data Structures, pp. 437–449 (1989)
27. Sion, R.: Query execution assurance for outsourced databases. In: VLDB '05, pp. 601–612. VLDB Endowment (2005)
28. Tamassia, R., Triandopoulos, N.: On the cost of authenticated data structures. Technical report, Brown University (2003)
29. Tamassia, R.: Authenticated data structures. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, Springer, Heidelberg (2003)
30. online database Zoho Creator. http://creator.zoho.com

Location Privacy Protection Through Obfuscation-Based Techniques

C.A. Ardagna, M. Cremonini, E. Damiani,
S. De Capitani di Vimercati, and P. Samarati

Dipartimento di Tecnologie dell'Informazione
Università di Milano – 26013 Crema - Italy
{ardagna,cremonini,damiani,decapita,samarati}@dti.unimi.it

Abstract. The widespread adoption of mobile communication devices combined with technical improvements of location technologies are fostering the development of a new wave of applications that manage physical positions of individuals to offer location-based services for business, social or informational purposes. As an effect of such innovative services, however, privacy concerns are increasing, calling for more sophisticated solutions for providing users with different and manageable levels of privacy. In this work, we propose a way to express users privacy preferences on location information in a straightforward and intuitive way. Then, based on such location privacy preferences, we discuss a new solution, based on obfuscation techniques, which permits us to achieve, and quantitatively estimate through a metric, different degrees of location privacy.

1 Introduction

Information regarding physical locations of individuals is rapidly becoming easily available for processing by online and mobile *Location-Based Services* (LBSs). Customer-oriented applications, social networks and monitoring services can be functionally enriched with data reporting where people are, how they are moving or whether they are close by specific locations. To this end, several commercial and enterprise-oriented LBSs are already available and have gained popularity [4]. Key to those new LBSs are modern location technologies that have reached good precision and reliability at costs that most people (e.g., the cost of mobile devices) and companies (e.g., the cost of integrating location technologies in existing telecommunication infrastructures) can economically sustain.

Combined with novel application opportunities, however, threats to personal privacy are ramping up [4], as witnessed by recent security incidents targeting privacy of individuals, revealed faulty data management practices, and unauthorized trading of users personal information (including ID thefts and unauthorized profiling). Location information is not immune from such threats and presents new dangers such as stalking or physical harassment.

In this scenario, a novel contribution of the paper is represented by a comprehensive solution aimed at preserving location privacy of individuals through artificial perturbations of location information collected by sensing technology.

In particular, location information of users is managed by a trusted middleware [5,6,9], which enforces users privacy through obfuscation-based techniques.

Key to this work is the concept of *relevance* as the adimensional metric for the location accuracy. A relevance value is always associated with locations and it quantitatively characterizes the degree of privacy artificially introduced into a location measurement. Based on relevance, it is possible to strike a balance between the need of service providers, requiring a certain level of location accuracy, and the need of users, asking to minimize the disclosure of personal location information. Both needs can be expressed as relevances and either quality of online services or location privacy can be adjusted, negotiated or specified as contractual terms.

The remainder of this paper is organized as follow. Section 2 presents related work. Section 3 discusses our working assumptions. Section 4 illustrates our approach for defining location privacy preferences and introduces the concept of *relevance*. Section 5 presents our obfuscation techniques. Section 6 describes a first solution to their composition and presents some examples of application. Section 7 gives our conclusions.

2 Related Work

Location privacy issues are the subject of growing research efforts. The main branch of current research on LBS privacy focuses on users anonymity or partial identities [6,7,9]. Beresford and Stajano [6] present *mix zones*, a method used to enhance privacy in LBSs managed by trusted middlewares. The solution is based on preset physical zones where all users are indiscernible from one another. However, this solution is suitable for services that track users movement rather than, as in our case, for services requiring a user location at a specific time. Bettini et. al. [7] propose a framework in charge of evaluating the risk of sensitive location-based information dissemination, and a technique aimed at supporting k-anonymity [14]. Gruteser and Grunwald [9] define k-anonymity in the context of location obfuscation and propose a middleware architecture and an adaptive algorithm for adjusting location information resolution according to anonymity requirements.

Other works study the possibility of protecting users privacy through the definition of complex rule-based policies [10,11]. Although policies-based solutions are suitable for privacy protection, users, often, are not willing to directly manage complex policies and, hence, refuse participation in pervasive environments. By contrast, in this work we have implemented a solution for expressing privacy preferences that is simple and intuitive.

Finally, the line of research closest to our work consists in the adoption of obfuscation techniques aimed at location privacy protection. Location obfuscation is complementary to anonymity. In particular, rather than anonymizing users identities, obfuscation-based solutions assume the identification of users and introduce perturbations into collected locations to decrease their accuracy. Duckham and Kulik [8] develop an obfuscation technique for protecting location privacy by artificially inserting into measurements some fake points with the same probability as the real user position. The paper proposes a formal framework providing a mechanism for balancing between user needs for high-quality information services and for location privacy. The work of Bellavista et al. [5] is based on points of interest with symbolic location granularity (e.g., city, country). This forces the privacy level to some predefined choices only, resulting in an excessively rigid solution.

Current obfuscation-based solutions have some shortcomings that our proposal tries to address. First, they do not provide a quantitative estimation of the actual privacy level, which makes them highly dependent on the application contexts and difficult to integrate into a full fledged location-based application scenario [1,3]. Next, just a single obfuscation technique is usually implemented. By contrast, our work introduces the concept of relevance as an adimensional metric for location accuracy, defines more obfuscation techniques and demonstrate the benefits of their composition.

3 Working Assumptions

Our work is based on two working assumptions that simplify our analysis with no loss of generality. Our first working assumption concerns the shape of a location measurement: *the area returned by a location measurement is planar and circular.* User location information, in fact, is affected by an intrinsic measurement error introduced by sensing technologies, resulting in spatial areas rather than geographical points. This assumption represents a particular case of the general requirement of considering convex areas and a good approximation for actual shapes resulting from many location technologies (e.g., cellular phones location). According to this assumption, a location measurement is defined as follows.

Definition 1 (Location measurement). *A location measurement of a user u is a circular area $Area(r, x_c, y_c)$, centered on the geographical coordinates (x_c, y_c) and with radius r, which includes the real user's position (x_u, y_u) with probability $P((x_u, y_u) \in Area(r, x_c, y_c)) = 1$.*

Definition 1 comes from observing that sensing technologies based on cellular phones usually guarantee that the real user's position falls within the returned area.

To discuss the effects of obfuscation techniques, we introduce our second assumption. Consider a random location within a location measurement $Area(r, x_c, y_c)$, where a "random location" is a neighborhood of random point $(\hat{x}, \hat{y}) \in Area(r, x_c, y_c)$. Our second assumption is that the probability that the real user's position (x_u, y_u) belongs to a neighborhood of a random point $(\hat{x}, \hat{y})$ is uniformly distributed over the whole location measurement. Accordingly, the joint probability density function (pdf) of the real user's position can be defined as follows.

Definition 2 (Uniform joint pdf). *Given a location measurement Area* (r, x_c, y_c)*, the joint probability density function (joint pdf)* $f_r(x, y)$ *of real user's position* (x_u, y_u) *to be in the neighborhood of point* (x, y) *is:*

$$f_r(x,y) = \begin{cases} \frac{1}{\pi r^2} & \textit{if } x, y \in Area(r, x_c, y_c) \\ 0 & \textit{otherwise.} \end{cases}$$

Same assumption can be found in other works on this topic [12]. In this work, assuming a uniform distribution simplifies the discussion with no loss of generality. Considering Gaussian-like distributions, the consequence on our work is that obfuscating simply by scaling the radius of a location measurement is ineffective, while stronger obfuscation effects can be still achieved by combining different techniques.

4 Privacy Preferences and Location Relevance

The ultimate goal of this work is to design a solution able to manage location privacy as a functional term, required or adjusted by users according to their preferences and application context, and negotiated as a service attribute by users and LBSs. To this end, location privacy should be measured and quantified with regard to the *accuracy* of a user position, that is, the more accurate the position, the less privacy. Furthermore, location privacy should be measured regardless of specific application contexts and should be expressed quantitatively as a service parameter without sticking to some coarse-grained preset meta-locations such as "city" or "department", which represents simplified instances of privacy preferences that should be supported by a most general and flexible solution.

Before defining obfuscation techniques and their combination, we discuss some key aspects: accuracy estimations of available location technologies, the specification of users privacy preferences, and the concept of relevance.

4.1 Location Accuracy and Measurement Quality

The accuracy of a location measurement necessarily depends on the specific sensing technology and on the environmental conditions. Several works describe available sensing technologies discussing their accuracy. In [15], the authors provide a survey of standard positioning solutions for cellular networks such as, *E-OTD* for GSM, *OTDOA* for Wideband CDMA (WCDMA), and *Cell-ID*. Specifically, E-OTD location method is based on the existing observed time difference (OTD) feature of GSM systems. The accuracy of the E-OTD estimation, in recent studies, has been found to range from 50m to 125m. Observed Time Difference Of Arrival (OTDOA), instead, is designed to operate over wideband-code division multiple access (WCDMA) networks. The positioning process achieves a location accuracy of 50m at most. Finally, Cell-ID is a simple positioning method based on cell sector information, where cell size varies from 1-3km in urban areas to 3-20km in suburban/rural areas.

To evaluate the quality of a given location measurement, its accuracy must be compared with the nominal accuracy that the adopted sensing technology can reach. To this end, we call r_{meas} the radius of a measured area and r_{opt} the radius of the area that would be produced if the best accuracy is reached. In other words, r_{meas} represents the *actual measurement error*, while r_{opt} is the *minimum error*. Therefore, the ratio r_{opt}^2/r_{meas}^2 is a good estimation of the quality of each location measurement. For instance, assume that a user position is located with accuracy r_{meas}=62.5m using E-OTD method, accuracy r_{meas}=50m using OTDOA, and accuracy r_{meas}=1km using Cell-ID. Optimal accuracy is r_{opt}=50m. Then, according to r_{opt}^2/r_{meas}^2, the area provided by OTDOA has a measurement quality of 1, whereas the others have a quality proportionally reduced to 0.8 for E-OTD, and 0.05 for Cell-ID.

4.2 User Privacy Preferences

Systems that want to let users express their privacy preferences must strike a balance between the two traditionally conflicting requirements of usability and expressiveness. Complex policy specifications, fine-grained configurations and explicit technological details discourage users from fully exploiting the provided functionalities. Our goal is then to allow users to express privacy preferences in an intuitive and straightforward way. Our solution is based on users privacy preferences specified as a *minimum distance* [8,13]. According to this setting, for example, a user can define "100 meters" as her privacy preference, which means that she demands to be located with an accuracy not better than 100 meters. Considering circular areas, the privacy requirement is implemented by enlarging the radius of the original measurement to 100 meters, at least. However, privacy preferences expressed as a minimum distance have the drawback of being meaningful if associated with a technique that enlarge the original measurement only. Another issue that is often neglected by traditional location obfuscation solutions is the possibility to compose different obfuscation techniques to increase their robustness with respect to possible de-obfuscation attempts performed by adversaries.

Therefore, a major challenge is to design a system able to integrate several obfuscation techniques still relying on the definition of privacy preference in its simplest form, e.g., it would be unrealistic to explicitly ask user to specify a particular composition of techniques. Our solution transforms a simple preference like a minimum distance into a more general functional term with the constraint that the final obfuscated area produced by different obfuscation techniques must be equivalent, in terms of location privacy, to the area that would be derived by just enlarging the radius of the original measurement to the specified minimum distance. This way, we let users specify their preferences in the most intuitive way, whereas we can adopt obfuscation techniques different and more robust than the simple radius enlargement. The availability of a single obfuscation technique by enlarging the radius, in fact, gives to an adversary the possibility of guessing a better user position by simply reducing the observed area. Our solution, instead, introduces additional obfuscation techniques, and therefore improves user privacy.

To this end, we first introduce the attribute λ that represents a *relative privacy preference* (or, in other terms, a relative degradation of the location accuracy). λ must be derived from the minimum distance specified by a user, which we call r_{min}, and from the radius of the original measurement, the previously introduced r_{meas}. Having assumed circular areas, the relative accuracy degradation obtained by setting r_{min} is:

$$\lambda = \frac{max(r_{meas}, r_{min})^2 - r_{meas}^2}{r_{meas}^2} = \frac{max(r_{meas}, r_{min})^2}{r_{meas}^2} - 1 \tag{1}$$

The term $max(r_{meas}, r_{min})$ represents the special case of a minimum distance r_{min} smaller than the original r_{meas}. This is possible because the user is not aware of the actual accuracy of sensing technologies and the original measure could already satisfy the privacy preference by itself. Accordingly, the term λ is zero when the measurement accuracy (i.e., r_{meas}) already satisfies the user requirement (i.e., r_{min}) and no transformation to the original measurement is needed to satisfy privacy preferences. Otherwise, when this is not the case (i.e., $r_{min} > r_{meas}$), λ corresponds to various degrees of accuracy degradation, e.g., $\lambda = 0.2$ means 20% of degradation, $\lambda = 1$ means 100% of degradation and any value $\lambda > 1$ corresponds to a degradation greater than 100%.

Up to this point, the first benefit achieved by deriving λ from r_{min} and r_{meas} is that we can process a privacy preference as a relative degradation rather than the strictly dimensional and tightly coupled with the enlargement of the measured area r_{min}.

The next step is to introduce other obfuscation techniques and select them, individually or combined, to produce an obfuscated area that degrades the original accuracy as imposed by λ. This way, we can employ an enriched set of obfuscation techniques still relying on the simple definition of r_{min} as the user privacy preference. The drawback, which we consider acceptable, is that we are changing the meaning of the user preference r_{min}, which is not necessarily the radius of the obfuscated area. Instead, it represents a logical constraints that can be informally expressed as: *the location area produced by one or more a priori undetermined obfuscation techniques must be equivalent, in term of privacy, to the one produced by enlarging the radius of the original measurement up to* r_{min}.

4.3 Relevance

Key to our work is the notion of *relevance*, defined as an adimensional, technology-independent metric for the accuracy of an obfuscated area. The relevance metric is a value $\mathcal{R} \in (0, 1]$ that tends to 0 when location information must be considered unreliable for application providers; it is equal to 1 when location information has best accuracy; and a relevance value in (0,1) corresponds to some degrees of accuracy. Accordingly, the *location privacy* provided by an obfuscated location is evaluated by (1-$\mathcal{R}$). The reason for choosing to represent the accuracy of a location as a primitive concept rather than the privacy is functional. We assume that LBSs have to manage locations that, on the one side, could be perturbed for privacy reasons, while on the other side could be required to have an accuracy not below a threshold to preserve a certain quality of service. In our solution, all locations have an associated relevance attribute, from an initial location affected by a measurement error of sensing technologies to all possible subsequent manipulations to provide privacy. This way, relevance is the general functional term that qualifies a location with respect to either accuracy or privacy requirements. Two important relevance values characterize our privacy management solution:

– *Initial relevance ($\mathcal{R}_{Init}$).* The metric for the accuracy of a user location measurement as returned by a sensing technology. This is the initial value of the relevance that only depends on the intrinsic measurement error.
– *Final relevance ($\mathcal{R}_{Final}$).* The metric for the accuracy of the final obfuscated area produced by satisfying a relative privacy preference λ. It is derived, starting by the initial relevance, through the application of one or more obfuscation techniques.

A third relevance value is used when the combination of techniques will be discussed. It represents the *intermediate relevance*, denoted $\mathcal{R}_{Inter}$, derived by applying the first of two obfuscation techniques.

With regard to $\mathcal{R}_{Init}$, it evaluates the accuracy of the actual area returned by a specific location measurement. A good metric is the ratio of the area that would have been returned if the best accuracy was achieved (i.e., the one with radius r_{opt}) and the actual measured area (i.e., the one with radius r_{meas}). $\mathcal{R}_{Final}$ instead, is derived from $\mathcal{R}_{Init}$ by considering the relative privacy preference λ.

Definition 3 ($\mathcal{R}_{Init}$ and $\mathcal{R}_{Final}$). *Given a location measurement area of radius r_{meas} measured by a sensing technology, a radius r_{opt} representing the best accuracy of sensing technologies and a relative privacy preference λ, initial relevance $\mathcal{R}_{Init}$ and final relevance $\mathcal{R}_{Final}$ are calculated as:*

$$\mathcal{R}_{Init} = \frac{r_{opt}^2}{r_{meas}^2} \tag{2}$$

$$\mathcal{R}_{Final} = \frac{\mathcal{R}_{Init}}{\lambda + 1} \tag{3}$$

These definitions represent, respectively, our general forms of $\mathcal{R}_{Init}$ and $\mathcal{R}_{Final}$. By definition of λ (see (1)), the term $\frac{1}{\lambda+1}$ represents the degradation of the initial $\mathcal{R}_{Init}$ that satisfies the user privacy preference. The corresponding obfuscated area will be qualified by relevance $\mathcal{R}_{Final}$. In equation (3), substituting the term $\mathcal{R}_{Init}$ with equation (2) and term λ with equation (1), it results that $\mathcal{R}_{Final} = \frac{r_{opt}^2}{r_{min}^2}$, assuming $r_{min} > r_{meas}$ in (1). This represents the value of $\mathcal{R}_{Final}$ that corresponds to degrading the accuracy by λ, as for user's privacy preference.

5 Obfuscation Techniques

We now present three basic obfuscation techniques and their operators. Since there could be one or two obfuscation steps in our solution, we generically call $\mathcal{R}$ the relevance associated with the area to be obfuscated and $\mathcal{R}'$ the relevance of the obfuscated area. If only one obfuscation step is performed, then $\mathcal{R}$ =$\mathcal{R}_{Init}$ and $\mathcal{R}'$ =$\mathcal{R}_{Final}$. For two obfuscation steps, we have $\mathcal{R}$ =$\mathcal{R}_{Init}$ and $\mathcal{R}'$ =$\mathcal{R}_{Inter}$ for the first one, and $\mathcal{R}$ =$\mathcal{R}_{Inter}$ and $\mathcal{R}'$ =$\mathcal{R}_{Final}$ for the second one.

Furthermore, we employ *obfuscation operators* as a logical representation of the physical transformations realized by different obfuscation techniques: *i)* the `Enlarge` operator (E) degrades the accuracy of an initial location area by enlarging its radius; *ii)* the `Shift` operator (S) degrades the accuracy of an initial location area by shifting its center; and *iii)* the `Reduce` operator (R) degrades the accuracy of an initial location area by reducing its radius.

5.1 Obfuscation by Enlarging the Radius

Obfuscating a location measurement area by increasing its radius (see Fig. 1(a)) is the technique that most current solutions adopt. Obfuscation is a probabilistic effect provided by the decreasing of the joint probability density function (pdf), which we can express as $\forall r, r' \in \mathbb{R}^+, r < r' : f_r(x,y) > f_{r'}(x,y)$. The relevance $\mathcal{R}'$ can be derived from $\mathcal{R}$ by using the ratio of the associated pdf as the scalar factor:

$$\mathcal{R}' = \frac{f_{r'}(x,y)}{f_r(x,y)} \cdot \mathcal{R} = \frac{r^2}{r'^2} \cdot \mathcal{R}, \quad \textit{with } r < r' \tag{4}$$

Therefore, given two relevances, $\mathcal{R}$ and $\mathcal{R}'$, and the radius r of the initial area, an obfuscated area calculated with this technique has a final radius: $r' = r\sqrt{\frac{\mathcal{R}}{\mathcal{R}'}}$.

Finally, radius r' can be expressed as a function of λ: $r' = r\sqrt{\lambda + 1}$. This result is straightforward from equations (3) and (4) and reflects the definition of λ, which depends from r_{min} and assumes an obfuscation by enlarging the radius.

For instance, let the user privacy preference be r_{min}=1 km. Suppose that the location measurement of a user u has radius r_{meas}=0.5 km, and the optimal measurement accuracy is r_{opt}=0.4 km. Given this information, relevance $\mathcal{R}_{Init}$ associated with the location measurement, and relative privacy preference λ are calculated as $\mathcal{R}_{Init}=\frac{r_{opt}^2}{r_{meas}^2}$=0.64, and $\lambda = \frac{\max(r_{meas}, r_{min})^2}{r_{meas}^2} - 1$=3, respectively. Having calculated the relative privacy preference λ, $\mathcal{R}_{Final}$ is derived as $\mathcal{R}_{Final}=(\lambda+1)^{-1}\mathcal{R}_{Init}$=0.16. Now, the obfuscation by enlarging the radius is applied and the obfuscated area is derived by calculating $r' = r_{meas}\sqrt{\lambda + 1}$=1 km. Note that, since a single obfuscation by enlarging the radius is used, $r' = r_{min}$. However, this example shows the computations that have to be applied when a double obfuscation is used (see Section 6).

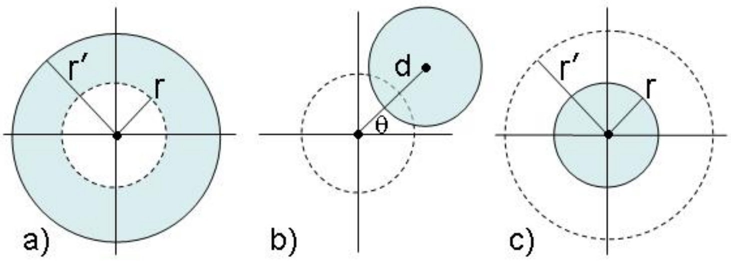


Fig. 1. Obfuscation by enlarging the radius (a), shifting the center (b), and reducing the radius (c)

5.2 Obfuscation by Shifting the Center

Shifting the center of a location measurement is another viable obfuscation technique (see Fig. 1(b)). An obfuscated area is derived from the original area by calculating the distance d between the two centers [2]. To measure the obfuscation effect and define the relation between relevances, two probabilities must be composed: *i)* the probability that the real user's position belongs to the intersection $Area_{Init \cap Final}$, and *ii)* the probability that a random point selected from the whole obfuscated area belongs to the intersection. Then, the relation between relevances $\mathcal{R}$ and $\mathcal{R}'$ is represented by:

$$\mathcal{R}' = P((x_u, y_u) \in Area_{Init \cap Final}) \cdot P((x, y) \in Area_{Init \cap Final}) =$$
$$\frac{Area_{Init \cap Final}}{Area(r, x_c, y_c)} \cdot \frac{Area_{Init \cap Final}}{Area(r, x_c + \Delta x, y_c + \Delta y)} = \frac{Area^2_{Init \cap Final}}{Area(r, x_c, y_c)^2} \cdot \mathcal{R} \qquad (5)$$

Recalling equations (3) and (5), it follows that $(\lambda + 1)^{-1} = \frac{Area^2_{Init \cap Final}}{Area(r, x_c, y_c)^2}$. Then, given λ, and πr^2 as the value of both areas, the overlapping can be expressed as: $Area_{Init \cap Final} = \pi r^2 / \sqrt{\lambda + 1}$.

Distance d between the centers is the unknown variable to be derived to obtain the obfuscated area. It can be calculated by expanding the term $Area_{Init \cap Final}$ as a function of d and by solving the following system of equations, whose variables are d, σ and γ. Here, σ and γ are the central angles of circular sectors identified by the two radii connecting the centers of the areas with the intersection points of original and obfuscated areas.[1]

$$\begin{cases} \left[\frac{\sigma}{2} r^2 - \frac{r^2}{2} \sin \sigma\right] + \left[\frac{\gamma}{2} R^2 - \frac{R^2}{2} \sin \gamma\right] = \sqrt{\delta} \pi r \cdot R \\ d = r \cos \frac{\sigma}{2} + R \cos \frac{\gamma}{2} \\ r \sin \frac{\sigma}{2} = R \sin \frac{\gamma}{2} \end{cases} \qquad (6)$$

[1] The system of equation (6) is presented in the most general form, where there are two areas with different radii (i.e., r and R).

Solutions of this system can be obtained numerically. By our definitions, obfuscated areas calculated by shifting the center satisfy a relative privacy preference λ and thus provides same privacy of an obfuscated area that would have been calculated with an enlarged radius r_{min}.

For instance, let the user specifies her privacy preference through r_{min}=1.42 km. Suppose that the location measurement of a user u has radius r_{meas}=1 km, and the optimal measurement accuracy is r_{opt}=0.8 km. Relevance $\mathcal{R}_{Init}$ and λ are calculated as $\mathcal{R}_{Init}$=0.64, and λ=1, respectively. Then, $\mathcal{R}_{Final}$=0.32 is derived and the obfuscation by shifting the center applied. At this point, distance d=0.464 km is calculated by solving the system of equation (6). Finally, an angle θ is randomly selected and the obfuscated area is generated.

5.3 Obfuscation by Reducing the Radius

The third obfuscation technique consists in reducing the radius r of one location from r to r', as showed in Fig. 1(c). The obfuscation effect is produced by a correspondent reduction of the probability to find the real user location within the returned area, whereas the joint pdf is fixed.

If we call (x_u, y_u) the unknown real user position coordinates, by assumption the probability that the real user position falls in the area of radius r is $P((x_u, y_u) \in Area(r, x, y)) = 1$. When we obfuscate by reducing the radius, an area of radius $r' \leq r$ is returned, where $P((x_u, y_u) \in Area(r', x, y)) \leq P((x_u, y_u) \in Area(r, x, y))$, since a circular ring having pdf greater than zero has been excluded.

With regard to relevances $\mathcal{R}$ and $\mathcal{R}'$, their relation can be defined as:

$$\mathcal{R}' = \frac{P((x_u, y_u) \in Area(r', x, y))}{P((x_u, y_u) \in Area(r, x, y))} \cdot \mathcal{R} = \frac{r'^2}{r^2} \cdot \mathcal{R}, \quad \text{with } r' < r \tag{7}$$

From (3) and (7), it follows that $(\lambda + 1)^{-1} = \frac{r'^2}{r^2}$. Then, given λ and r, the area returned when obfuscation by reducing the radius is applied has radius: $r' = \frac{r}{\sqrt{\lambda+1}}$. Again, similarly to the previous technique, obfuscated areas calculated in this way by reducing the radius satisfy, according to our semantics, a relative privacy preference λ and consequently, also the corresponding user privacy preference r_{min}.

For instance, let the user privacy preference be r_{min}=1 km. Suppose that the location measurement of a user u has radius r_{meas}=0.5 km, and the optimal measurement accuracy is r_{opt}=0.4 km. Relevance $\mathcal{R}_{Init}$ and λ are calculated as $\mathcal{R}_{Init}$=0.64, and λ=3. $\mathcal{R}_{Final}$ is derived as $\mathcal{R}_{Final}=(\lambda+1)^{-1}\mathcal{R}_{Init}$=0.16. Now, the obfuscation by reducing the radius is applied and the obfuscated area, respecting user privacy preference r_{min}, is derived by calculating $r' = \frac{r_{meas}}{\sqrt{\lambda+1}}$=0.25 km.

6 Double Obfuscation

Given the obfuscation techniques just introduced, users privacy preferences can be satisfied either by using one technique among the three or by composing two

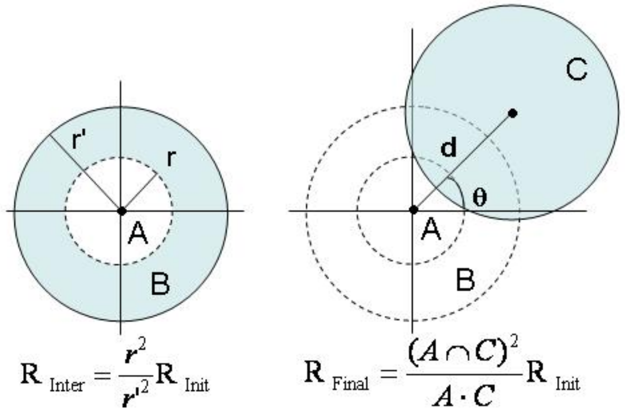

r'
r
A
B
$R_{Inter} = \frac{r^2}{r'^2} R_{Init}$
C
d
θ
A
B
$R_{Final} = \frac{(A \cap C)^2}{A \cdot C} R_{Init}$

Fig. 2. Example of $E - S$ obfuscation

techniques. In the last case, an obfuscated area produced by one obfuscation technique (operator h) is further obfuscated by the application of a second technique (operator g). Formally, let be $\mathcal{A}$ the set of location areas, $h : \mathcal{A} \rightarrow \mathcal{A}$ and $g : \mathcal{A} \rightarrow \mathcal{A}$ be two obfuscation operators, where the areas produced by applying the operator h are the inputs of the second operator g, which finally produces the obfuscated areas. Recalling that the ultimate goal of an obfuscation process is to reduce the location accuracy estimated by an initial relevance $\mathcal{R}_{Init}$ to a final relevance $\mathcal{R}_{Final}$, in case of double obfuscation, the intermediate term $\mathcal{R}_{Inter}$ must be introduce to represent the relevance achieved by the first obfuscation step.

As a special example of obfuscation composition, let us consider an enlargement of the radius followed by a shift of the center (see Fig. 2). For the first obfuscation, the radius of area B is calculated by applying operator `Enlarge` and equation $\mathcal{R}' = \frac{r^2}{r'^2} \cdot \mathcal{R}$, with $\mathcal{R} = \mathcal{R}_{Init}$ and $\mathcal{R}' = \mathcal{R}_{Inter}$. For the second step, operator `Shift` is used with an important constraint: the domain of the `Shift` operator must be restricted to those areas that have an intersection with the *original* measured area (i.e., in our example the intersection between the final obfuscated area C and area A should not be empty). As a consequence, to calculate the final obfuscated area C, we need to determinate distance d, which depends on the overlap between area A and C. The reason is that to respect privacy preference λ, the second operator of a composition must be always referred to the original measured area A. Accordingly, equation $\mathcal{R}' = \frac{(A \cap C)^2}{A \cdot C} \cdot \mathcal{R}$, has $\mathcal{R}' = \mathcal{R}_{Final}$ and $\mathcal{R} = \mathcal{R}_{Init}$, rather than $\mathcal{R} = \mathcal{R}_{Inter}$ as we would have expected in general.

Finally, we observe that, whereas in theory it is possible to compose operators E, R, and S in an indeterminate number of steps, there is never any convenience to combine more than two techniques. This follows by a geometric property of circles assuring that, given two circles A_1 and A_2, A_2 can be generated starting from A_1 through two geometric operations at most: one center-shifting, and one between radius enlargement or reduction. Finally, we observe that, as for most composable functions, the *commutative property* does not hold for the

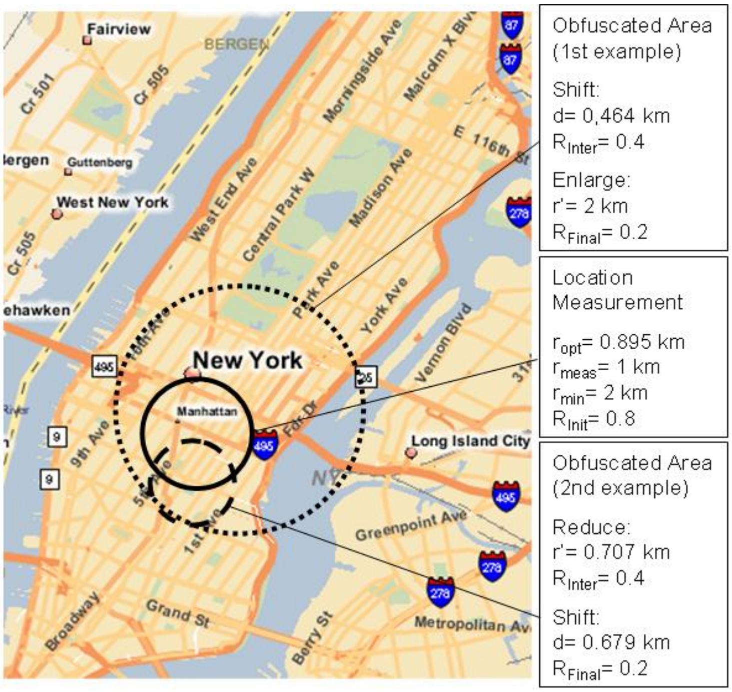


Fig. 3. Example of $S-E$ and $S-R$ obfuscation on a large scale

composition of operators `Enlarge` or `Reduce` with `Shift`. Therefore, the available obfuscation choices are: *i)* traditional single obfuscations E, R, and S; *ii)* double obfuscations $E-S$, $S-E$, $R-S$, and $S-R$.

6.1 Double Obfuscation Examples

We describe two examples of possible application of double obfuscation. For sake of clarity, we suppose an user located in Manhattan, around the *Empire State Building.* A location measurement is assumed to have radius r_{meas}=1km (see the area with filled line in Fig. 3).[2] The user has specified her privacy preference as r_{min}=2km. Given these information, λ, $\mathcal{R}_{Init}$, and $\mathcal{R}_{Final}$ are calculated before applying obfuscations.

For the first example, an $S-E$ obfuscation has been applied. The obfuscation process starts by setting $\theta = \pi/4$ and $\mathcal{R}_{Inter}$=0.4.

Distance d=0.464km is calculated by solving the system of equations (6), and location measurement area is shifted accordingly generating an obfuscated area of relevance $\mathcal{R}_{Inter}$. Finally, the `Enlarge` operator is applied to the area with relevance $\mathcal{R}_{Inter}$, and final radius r'=2 km is computed to achieve relevance

[2] In these examples, r_{opt}=0.895km. We are aware that this assumption is far from reality, but it was assumed for simplicity.

$\mathcal{R}_{Final}$. This way, when the user location is released to a LBS she results positioned in a bigger area that includes nearly all Central Manhattan (see the area with dotted line in Fig. 3).

For the second example, suppose a $R-S$ obfuscation. Again, for simplicity we set $\mathcal{R}_{Inter}$=0.4 and $\theta = 5\pi/3$. Radius r'=0.707 km is computed from (7) and the first obfuscated area is produced. Then, the system of equations (6) is solved numerically resulting in d =0.679 km, and the center is shifted. This way, when the user location is released to a LBS, the user seems located just around *Madison Square* (see the area with dashed line showed in Fig. 3).

7 Conclusions and Future Work

We presented privacy-enhanced techniques that protect user privacy based on spatial obfuscation. Our proposal aims at achieving a solution that both considers the accuracy of location measurements, which is an important feature of location information, and the need of privacy of users. In addition to several obfuscation techniques for privacy preservation, we also present and define a formal and intuitive way to express users privacy preferences, and a formal metric for location accuracy. Issues to be investigated include the analysis of our solution assuming Gaussian-like distributions, the evaluation of obfuscation techniques robustness against de-obfuscation attacks, and the possibility to manage different privacy preferences expressed by users.

Acknowledgments

This work was partially supported by the European Union within the PRIME Project in the FP6/IST Programme under contract IST-2002-507591, by the Italian Ministry of Research Fund for Basic Research (FIRB) under project RBNE05FKZ2 and by the Italian MIUR under project MAPS. We would like also to thank the anonymous referees for their helpful comments on the paper.

References

1. Ardagna, C.A., Cremonini, M., Damiani, E., De Capitani di Vimercati, S., Samarati, P.: Supporting location-based conditions in access control policies. In: Proc. of the ACM Symposium on Information, Computer and Communications Security (ASIACCS'06), Taipei, Taiwan (March 2006)
2. Ardagna, C.A., Cremonini, M., Damiani, E., De Capitani di Vimercati, S., Samarati, P.: Managing privacy in LBAC systems. In: Proc. of the Second IEEE International Symposium on Pervasive Computing and Ad Hoc Communications (PCAC-07), Niagara Falls, Canada (May 2007)
3. Atluri, V., Shin, H.: Efficient enforcement of security policies based on tracking of mobile users. In: Proc. of the 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, pp. 237–251, Sophia Antipolis, France (2006)

4. Barkhuus, L., Dey, A.: Location-based services for mobile telephony: a study of user's privacy concerns. In: Proc. of the 9th IFIP TC13 International Conference on Human-Computer Interaction (INTERACT 2003), pp. 709–712, Zurich, Switzerland (September 2003)
5. Bellavista, P., Corradi, A., Giannelli, C.: Efficiently managing location information with privacy requirements in wi-fi networks: a middleware approach. In: Proc. of the International Symposium on Wireless Communication Systems (ISWCS'05), pp. 1–8, Siena, Italy (September 2005)
6. Beresford, A.R., Stajano, F.: Mix zones: User privacy in location-aware services. In: Proc. of the 2nd IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMWO 04) (2004)
7. Bettini, C., Wang, X.S., Jajodia, S.: Protecting privacy against location-based personal identification. In: Jonker, W., Petković, M. (eds.) SDM 2005. LNCS, vol. 3674, Springer, Heidelberg (2005)
8. Duckham, M., Kulik, L.: A formal model of obfuscation and negotiation for location privacy. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468, Springer, Heidelberg (2005)
9. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: Proc. of the 1st International Conference on Mobile Systems, Applications, and Services (2003)
10. Hauser, C., Kabatnik, M.: Towards Privacy Support in a Global Location Service. In: Proc. of the IFIP Workshop on IP and ATM Traffic Management (WATM/EUNICE 2001), Paris, France (2001)
11. Langheinrich, M.: A privacy awareness system for ubiquitous computing environments. In: Borriello, G., Holmquist, L.E. (eds.) UbiComp 2002. LNCS, vol. 2498, pp. 237–245. Springer, Heidelberg (2002)
12. Mokbel, M.F., Chow, C-Y., Aref, W.G.: The new casper: Query processing for location services without compromising privacy. In: Proc. of the 32nd International Conference on Very Large Data Bases, pp. 763–774, Korea (2006)
13. Openwave. Openwave Location Manager (2006) `http://www.openwave.com/`
14. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering 13(6), 1010–1027 (2001)
15. Sun, G., Chen, J., Guo, W., Ray Liu, K.J.: Signal processing techniques in network-aided positioning: A survey of state-of-the-art positioning designs. IEEE Signal Processing Magazine, pp. 12–23, July (2005)

Efficient Security Policy Enforcement in a Location Based Service Environment[*]

Vijayalakshmi Atluri and Heechang Shin

MSIS Department and CIMIC, Rutgers University, USA
{atluri,hshin}@cimic.rutgers.edu

Abstract. Location based services, one of the promising markets of mobile commerce, aims at delivering point of need personalized information. Often, these services to be delivered are based on the prior knowledge of the profiles of mobile customers and security and privacy policies dictated by them. These policies may specify revealing the sensitive information of mobile customers (e.g., age, salary) selectively to specific merchants in return of receiving certain benefits (e.g., coupons, special discounts, etc.). As a result, the security policies in such an environment are characterized by spatial and temporal attributes of the mobile customers (location and time), as well as their profile attributes. The focus of this paper is to efficiently enforce such policies. In this regard, we propose a unified structure that is capable of indexing mobile customer (mobile object) locations and their profiles, and the authorizations stating their security and privacy policies.

1 Introduction

In recent years, mobile phones and wireless PDAs have evolved into wireless terminals that are Global Positioning System (GPS) enabled. With the expected revenues of mobile commerce to exceed \$88 billion by 2009 [12], mobile commerce will soon become a gigantic market opportunity. The market for location-aware mobile applications, often known as *location-based services* (LBS), is very promising. LBS is to request usable, personalized information delivered at the point of need, which includes information about new or interesting products and services, promotions, and targeting of customers based on more advanced knowledge of customer profiles and preferences, automatic updates of travel reservations, etc. For example, a LBS provider can be designed to present users with targeted content such as clothing items on sale, based on prior knowledge of their profile, preferences and/or knowledge of their current location, such as proximity to a shopping mall [13]. Additionally, LBS can provide nearby points of interests based on the real-time location of the mobile customer, advising of current conditions such as traffic and weather, deliver personalized, location-aware, and context-sensitive advertising, again based on the mobile customer profiles and preferences. Whether such LBS is delivered in a "push" or "pull" fashion, service providers require access to customers' preference profiles either through a proprietary database or use an arrangement with an LBS provider, who matches customer profiles to vendor offerings [11].

[*] This work is supported in part by the National Science Foundation under grant IIS-0242415 and Rutgers Business School.

In order to implement such services, customization and personalization based on the location information, customer profiles and preferences, and vendor offerings are required. This is because, to be effective, targeted advertising should not overwhelm the mobile consumers and must push information only to a certain segment of mobile consumers based on their preferences and profiles, and based on certain marketing criteria. Obviously, these consumers should be targeted only if they are in the location where the advertisement is applicable at the time of the offer. It is important to note here that user profile information may include both sensitive and non-sensitive attributes such as name, address, linguistic preference, age group, income level, marital status, education level, etc. Certain segment of mobile consumers are willing to trade-off privacy by sharing such sensitive data with selective merchants, either to benefit from personalization or to receive incentives offered by the merchants. Therefore, it is important that the sensitive profile information is revealed to the respective merchants only on the need-to-know basis. For example, a security policy may specify that a customer is willing to reveal his age in order to enjoy a 20% discount coupon offered on sports clothing. But he is willing to do this only during the evening hours and while close to the store. As a result, the security policies in such an environment are characterized by spatial and temporal attributes of the mobile customers (location and time), as well as their profile attributes.

A trusted LBS service provider can ensure that the customer's security policy is enforced without revealing the real identity of the customer to the merchant. Thus, an appropriate access control mechanism must be in place to enforce the authorization specifications reflecting the above security and privacy needs.

Traditionally, access policies are specified as a set of authorizations, where each authorization states if a given subject possesses privileges to access an object. Considering the basic authorization specification $\langle subject, object, privilege \rangle$, in a mobile environment, a moving object can be a subject, an object, or both. Access requests in such an environment can typically be on *past, present* and *future* status of the moving objects [2,3]. Serving an access request requires to search for the desired moving objects that satisfy the query, as well as enforce the security policies. The focus of this paper is to address the problem of efficiently enforcing such security policies.

Enforcing security policies often degrades the performance of a system. One way to alleviate the problem is to efficiently organize the mobile objects as well as authorizations. An index scheme for moving object data and user profiles has been proposed by Atluri et al. [7], but this does not consider authorizations. Recently, Atluri and Guo [4] have proposed a unified index structure called ^{S}TPR-tree in which authorizations are carefully overlaid on a moving object index structure (TPR-tree) [6], based on their spatiotemporal parameters. One main limitation of the ^{S}TPR-tree is that it is not capable of maintaining past information. As a result, it cannot support queries based on past location and security policies based on $tracking$ of mobile users. More recently, Atluri and Shin [5], present an index structure, called S^{PPF}-tree, which maintains past, present and future positions of the moving objects along with authorizations by employing the *partial persistent storage*. An index structure has been proposed to index authorizations ensuring that the customer profile information be disclosed to the merchants based on the choice of the customers [1]. However, this provides separate index structures for data and authorizations, and therefore is not a unified index. In essence, none of the previously proposed unified indexing schemes support security policy enforcement based on the profiles of the mobiles users.

In this paper, we present an index structure, called S^{STP}-tree, which maintains authorizations along with present and future locations as well as profiles of moving objects. In particular, we build on the concepts of the TPR-tree [6] and modify the node structure of the tree to hold profile information. Then, authorizations are overlaid suitably on the nodes of the tree. In order to support the profile information, we propose a Profile Bounding Vector (PV^B) which works similar to Minimum Bounding Rectangle (MBR) in the R-tree family. We demonstrate how the S^{STP}-tree can be constructed and maintained, and provide algorithms to process access requests. Our analysis shows that under normal circumstances, S^{STP}-tree performs better than utilizing two separate indexes (one for moving objects and another for profile). More specifically, if the data size of a PV^B is small enough so that the minimum number of the data (or children) that a leaf node (or non-leaf node) is the same between the S^{STP}-tree and the separate index structures, the analysis shows that our tree performs better.

This paper is organized as follows. Section 2 introduces preliminaries such as user profiles and the TPR-tree. In section 3, moving object authorization model is presented, and in section 4, we present our approach, called S^{STP}-tree, and section 5 illustrates our approach and strategy to evaluate user requests. Also, theoretical analysis is presented. In section 6, we conclude the paper by providing some insights into our future research in this area.

2 Preliminaries

In this section, we present the preliminary concepts on building an index structure for moving objects and profiles.

2.1 Moving Objects

Representation of Moving Objects: Let the set of moving objects be O=$\{o_1,o_2,\ldots,o_k\}$. In the d-dimensional space, objects are specified as points which move with constant velocity $\bar{v} = \{v_1, v_2, \ldots, v_d\}$ and initial location $\bar{x} = \{x_1, x_2, \ldots, x_d\}$. The position $\bar{x}(t)$ of an object at future time $t(t \geq t_c)$ can be computed through the linear function of time, $\bar{x}(t) = \bar{x}(t_0) + \bar{v}(t - t_0)$ where t_0 is the initial time, t_c the current time and $\bar{x}(t_0)$ the initial position. Considering a two-dimensional space, a moving object o_i moving in $\langle x, y\rangle$ space can be represented as follows: $o_i = ((x_i, v_{i_x}), (y_i, v_{i_y}))$.

Time Parameterized Bounding Rectangle (*tpbr*)**:** Given a set of moving objects O = $\{o_1, \ldots, o_n\}$ in the time interval $[t_0, t_0 + \delta t]$ in $\langle x, y, t\rangle$ space, the *tpbr* of O is a 3-dimensional bounding trapezoid which bounds all the moving objects in O during the entire time interval $[t_0, t_0 + \delta t]$ in the following way:

$$tpbr(O) = \{(x^{\vdash}, x^{\dashv}, y^{\vdash}, y^{\dashv}), (v_x^{\vdash}, v_x^{\dashv}, v_y^{\vdash}, v_y^{\dashv})\}\ where\, \forall\, i \in \{1, 2, \ldots, n\}$$

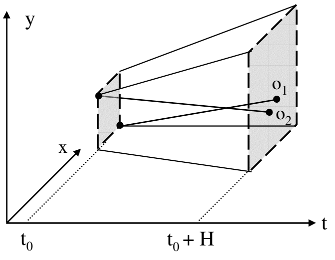


Fig. 1. The Time Parameterized Bounding Rectangle (*tpbr*)

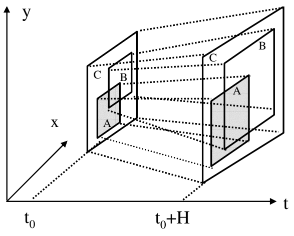


Fig. 2. The *tpbr* Hierarchy

$$x^{\vdash} = x^{\vdash}(t_0) = min_i\{x_i(t_0)\} \qquad v_x^{\vdash} = min_i\{v_{i_x}\}$$
$$x^{\dashv} = x^{\dashv}(t_0) = max_i\{x_i(t_0)\} \qquad v_x^{\dashv} = max_i\{v_{i_x}\}$$
$$y^{\vdash} = y^{\vdash}(t_0) = min_i\{y_i(t_0)\} \qquad v_y^{\vdash} = min_i\{v_{i_y}\}$$
$$y^{\dashv} = y^{\dashv}(t_0) = max_i\{y_i(t_0)\} \qquad v_y^{\dashv} = max_i\{v_{i_y}\}$$

Then, we can compute the bounding rectangles that *tpbr* covers with respect to time. The bounding rectangle's x-axis interval and y-axis interval at time t are defined as $[x^{\vdash}(t), x^{\dashv}(t)] = [x^{\vdash}(t_0) + v_x^{\vdash}(t - t_0), x^{\dashv}(t_0) + v_x^{\dashv}(t - t_0)]$ and $[y^{\vdash}(t), y^{\dashv}(t)] = [y^{\vdash}(t_0) + v_y^{\vdash}(t - t_0), y^{\dashv}(t_0) + v_y^{\dashv}(t - t_0)]$ respectively.

Time Horizon (H): Given a moving object, it is unrealistic to assume that its velocity remains constant. Therefore, the predicted future location of a object specified as a linear function of time becomes less and less accurate as time elapses [6]. To address this issue, a *time horizon* H is defined, which represents the time interval during which the velocities of the moving objects assumed to be the same. Figure 1 shows how *tpbr* bounds the trajectory of two moving objects o_1 and o_2 in $[t_0, t_0 + H]$.

The Tree Structure: Given a set of *tpbr*s, they can be organized in a hierarchical structure. In figure 2, *tpbr* C encloses *tpbr*s A and B. These three can be organized as a hierarchical structure with A and B being the children of C, Essentially, at the bottom-most level of the hierarchy, a set of moving objects could be grouped to form *tpbr*s. Each *tpbr* of the next higher level is the bounding *tpbr* of the set of *tpbr*s of all of its children. The root of the hierarchy is thus the bounding *tpbr* covering all its lower level *tpbr*s in a recursive manner.

2.2 User Profiles

We assume user profile as a set of attributes associated with a mobile customer that characterizes the user. These attributes may include (1) demographic information (e.g. country, race, age, gender, etc.), (2) contact information (e.g., name, address, zip code,

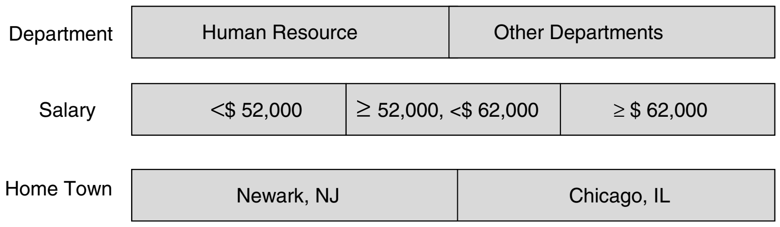


Fig. 3. Profile Attribute Discretization

telephone number, e-mail, etc.), (3) personal preferences (e.g., hobbies, favorite activities, favorite magazines, etc.), and (4) behavioral profile (e.g., level of activity, type of activity, etc.)[1]

Let the set of profile attributes under consideration be $P = \{p_1, p_2, \ldots, p_n\}$. We assume the profile of each user u is $\{p_1 : val_1, p_2 : val_2, \ldots, p_n : val_n\}$, where val_i is the corresponding value of p_i for that user. Since all attributes are not applicable to all users, some of these attributes may be empty for certain users.

Representation of User Profile: Given a profile attribute p_i, we first discretize it if necessary. We simply use as many bits as the number of all the possible discrete values for p_i. If the attribute is numerical data type, we partition the continuous data space into disjoint, mutually-exclusive intervals, as shown for attribute age in figure 3. The details of discretization method can be found in [14].

Definition 1 (Profile Vector). *Given a profile of user* $u = \{p_1 : val_1, p_2 : val_2, \ldots, p_n : val_n\}$, *we define a profile vector of* u, *denoted as* pv_u *as follows:* $pv_u = \{v_1, v_2, \ldots v_n\}$, *where each* v_i *is a sequence of binary digits such that the number of digits is equal to the number of discrete values of* p_i, *and the digits is 1 if* val_j *satisfies the corresponding discrete value, and 0 otherwise.*

Table 1. User Profile Information

Name	*Department*	*Salary*	*Home Town*	*Profile Vector*
Doe	Human Resource	\$63,000	Newark, NJ	$\langle 10, 001, 10 \rangle$
James	Other Departments	\$45,000	Chicago, IL	$\langle 01, 100, 01 \rangle$
Robert	Human Resource	\$53,000	Chicago, IL	$\langle 01, 010, 01 \rangle$

Figure 3 presents how each profile attribute can be represented. For example, because all the possible values of a profile attribute, *Department* are two ("Human Resource" and

[1] The behavioral profile is created by observing activities and habits of a user continuously. For example, Sony TiVo box records frequently-watched television shows and generates a behavioral profile based on the past patterns. In order to do so, information such as what kind of activity has been done by a user at what intensity needs to be captured. In case of TiVo, type of activity can be 'watching drama' and level of activity can be '2 hours.'

”Other Departments”), we use two bits to represent $Department$: ”Human Resource” is represented with ’10’ and ”Other Departments” is represented with ’01.’ Also, if $Salary <$ \$52,000, we represent it with ’100’, and ’001’ if $Salary \geq$ \$62,000. Table 1 shows the examples of user profile vectors. For example, profile representation of the user, Doe, is $\langle 10, 001, 10 \rangle$ because his department ’Human Resource’, is represented as ’10’, salary, \$63,000, as 001, and home town, ’Newark, NJ’ as ’10’.

3 Moving Object Authorization Model

In this section, we review the authorization model presented in [4] that is capable of specifying access control policies based on the spatial and temporal attributes, and suitably extend it to specify policies based on the profile information of mobile users.

Definition 2 (Authorization). *An* authorization α *is a 4 tuple* $\langle se, ge, m, \tau \rangle$*, where* se *is a* subject expression *denoting a set of subjects,* ge *is a* object expression *denoting a set of objects,* m *is a set of privilege modes, and* τ *is a temporal term.* □

The formalism to specify se[2], ge and τ has been developed in [8]. Due to space limitations, we do not review these details in this paper. Subject expression se can be used to specify a set of subjects such that they are associated with (i) a set of spatiotemporal and/or other traditional credential attributes and/or profile attributes, (ii) a set of subject identifiers, or (iii) a combination of both. In the same way, object expression ge can be used to specify a set of objects such that they are associated with (i) a set of spatiotemporal and/or other types of attributes, (ii) a set of object identifiers, or (iii) a combination of both.

Note that the set of subjects and objects denoted by se and ge can be moving objects. Because both a subject and an object can be a moving object, to avoid confusion, from now on, we denote the objects specified in the authorization as *auth-objects* (stands for authorization objects). In a LBS environment, generally subjects are moving objects, and location-aware information (such as near-by restaurants, route management, and so on) is provided to subjects based on their location. A policy may state that a (moving) subject may access an auth-object such as files, printers based on the subject’s spatiotemporal and profile attributes. Additionally, in a security policy, the (stationary) subject is allowed to access the profile information associated with a (moving) object.

The privilege mode m[3] supports not only read, write, and execute privileges for traditional auth-objects but also viewing, locating, tracking, send_sms_message for moving objects. A temporal term τ can be a time point, a time interval or a set of time intervals.

For a given authorization $\alpha = \langle se, ge, m, \tau \rangle$, we denote subjects expressed by se as $\alpha.se$, objects expressed by ge as $\alpha.ge$, privileges as $\alpha.m$, and temporal term $\alpha.\tau$, respectively.

There are two types of data associated with a moving object: spatiotemporal data (such as location) and non-spatiotemporal data (such as the profile information). Profile

[2] In [8], ce was used instead of se.

[3] Typically, the set of privileges M forms a partial order, where the ordering relationship can be represented with $\prec_m$.

information denotes a set of attributes that describes a moving object. For example, a person with handheld tracking devices can be described by her name, age, salary, occupation, and so on. Therefore, the security policies must be able to specify under what conditions, a mobile user wants to reveal her sensitive information. As such, given an authorization α, a subject expression se and an object expression ge, it is possible that the attributes involved in both se and ge could either be spatiotemporal in nature or not. Therefore, we extract the spatiotemporal and non-spatiotemporal part of the expression in α and denote them as $\alpha^{\Box}$ and $\alpha^{\rightarrow}$, respectively. Note that, not all profile attributes of a mobile customer are relevant. We consider only those attributes that are involved in the entire set of policies to be included in the profile vector.

If se includes the specification of spatiotemporal region, it means that the moving subjects specified in se must be within the region to gain access to the objects specified in ge. Similarly, only the mobile objects in se that are within the region are allowed to be accessed by the subjects specified in se, if ge includes the specification of spatiotemporal region. Thus, an authorization embeds a spatiotemporal region for specifying authorizing conditions for se or ge.

In the following, we present some examples of security policies.

- Policy 1: In order to get a personalized promotion deal, a mobile customer is willing to reveal her age and salary information to a merchant, provided she is within 10 miles from the shopping mall during evening hours. This policy can be expressed as follows.
  α_1 = $\langle$merchant(i), {customer(j) $\wedge$ location(j)=circle((50,60),10) $\wedge$ [5pm, 9pm] $\wedge$ age(j) $\wedge$ salary(j) }, sms_message $\rangle$
  Here, only the attributes involved in ge are spatiotemporal in nature, but not those involved in se. In the above, $\alpha_1^{\Box}$ is represented by a circle centered at (50,60) with radius 10 miles and t-axis interval = [5pm, 9pm]. Also, $\alpha_1^{\rightarrow}$ = $\langle **, ***, ** \rangle$ because policy 1 does not evaluate the profiles of mobile users.
- Policy 2: When a mobile user enters the Newark Liberty International Airport, taxi companies are allowed to access her current location information if she is within the airport during office hours.
  α_2 = $\langle$taxi_company(i), {customer(j) $\wedge$ location(j)=rectangle($10, 70, 2, 2$) $\wedge$ [9am, 5pm] }, locate$\rangle$[4]
  Here, only the attributes involved in ge are spatiotemporal in nature. $\alpha_2^{\Box}$ is represented by a three dimensional rectangle with x-axis interval = [10, 12], y-axis interval = [70, 72], and t-axis interval = [9am, 5pm]. $\alpha_2^{\rightarrow}$ = $\langle **, * * *, 01 \rangle$ because α_2 evaluates only the profile attribute 'Home Town'.
- Policy 3: Any employee can send a print job if she is currently located at the office during the office hours.
  α_3 = $\langle$emp(i) $\wedge$ rectangle(i)=(3,5,1,2) $\wedge$ [9am, 5pm], printer(j), write $\rangle$
  Here, only the attributes involved in se are spatiotemporal in nature, but not those involved in ge. $\alpha_3^{\Box}$ is represented by a three dimensional rectangle with x-axis

[4] Services such as the *gazetteer service* [9] that convert canonical geographic area names into geo-coordinates are available today. For example, a place name such as "Newark Airport, NJ" can be converted into the coordinates as shown above.

interval = [3, 4], y-axis interval = [5, 7], and t-axis interval = [9am, 5pm]. Also, $\alpha_3^{\rightarrow}$ = $\langle **, ***, ** \rangle$ because policy 3 does not evaluate the profiles of mobile users.

– Policy 4: A human resource employee is allowed to access performance records of employees only during office hours and while he is physically in his office.
  α_4= $\langle$ emp(i) $\wedge$ department(i) = 'human_resource' $\wedge$ rectangle(i)=(3,5,1,2) $\wedge$ [9am, 5pm] }, {performance_record(j)}, read $\rangle$
  Only the attributes involved in se are spatiotemporal in nature, but not those involved in ge. $\alpha_4^{\square}$ is represented by a three dimensional rectangle with x-axis interval = [3, 4], y-axis interval = [5, 7], and t-axis interval = [9am, 5pm]. Also, $\alpha_4^{\rightarrow}$ = $\langle 10, ***, ** \rangle$ because α_4 evaluates only the profile attribute 'Department'.

One main characteristic of authorization in the mobile environment is that the corresponding subject and object for se and ge are dynamically defined based on their location. For example, in a given authorization α, if $\alpha.ge$ specifies a region of Newark, NJ, $\alpha.se$ is authorized to access all the moving objects that lie within Newark, NJ to perform the privilege $\alpha.m$ on them. Thus, as time elapses, the list of authorized objects are being changed depending their movement inwards or outwards to the authorized area.

4 Unified Index Scheme for Moving Objects

In this section, we introduce our novel unified index structure, called the S^{STP}-tree that supports efficient enforcement of security policies based on present user locations as well as profiles. S^{STP}-tree is a balanced tree. Each node in the S^{STP}-tree comprises of the spatiotemporal attributes as well as a profile bounding vector, denoted as PV^B (explained later), in order to support the profile conditions. PV^B works similar to MBR (Minimum Bounding Rectangle) in R-tree. MBR in R-tree works as a coarse spatial filter that is used as a pre-filter to perform a more computationally expensive overlapping polygon checking [10]. Similarly, the role of profile bounding vector is to filter out profile conditions that do not satisfy the designated profile query conditions.

Profile Bounding Vector: In the following, we define a bounding vector that covers a set of profile vectors belonging to a set of users.

Definition 3 (Profile Bounding Vector). *Given a set of profile vectors PV* = $\{pv_1, pv_2, \ldots pv_n\}$, *such that each* $pv_i = \{v_{i1}, v_{i2}, \ldots v_{im}\}$. *A profile bounding vector of PV, denoted as* $PV^B = \{\{v_{11} \vee v_{21} \ldots v_{n1}\}, \{v_{12} \vee v_{22} \ldots v_{n2}\}, \ldots \{v_{1m} \vee v_{2m} \ldots v_{nm}\}\}$.

Let us consider once again the example in section 3 to explain the concept of PV^B. The set of profile attributes is department, salary, and home town. Consider the three profile vectors, pv_{Doe} = $\langle 10, 001, 10 \rangle$, pv_{James} = $\langle 01, 100, 01 \rangle$, and pv_{Robert} = $\langle 01, 010, 01 \rangle$. Then, PV^B of two users, Doe and James is $\langle 11, 101, 11 \rangle$, and PV^B of all three users is $\langle 11, 111, 11 \rangle$.

Given a set of PV^Bs, hierarchical structure can be formed. Suppose we have three PV^Bs.

$$PV_1^B = \langle 11, 011, 10 \rangle$$
$$PV_2^B = \langle 10, 010, 10 \rangle$$
$$PV_3^B = \langle 01, 001, 10 \rangle$$

These PV^Bs can be organized in a hierarchical structure with PV_2^B and PV_3^B as the children of PV_1^B. Each PV^B bounds PV^Bs of all of its children. Therefore, the root of the hierarchy covers the set of PV^Bs of all of its descendants.

Construction of S^{STP}-Tree: S^{STP}-tree is constructed similar to that of the tree structure described in section 2.1, but PV^B is updated accordingly during the insertion of new objects. As discussed in section 2, each moving object is represented with its spatiotemporal and profile attributes. Thus, each node in the S^{STP}-Tree includes a $tpbr$ and a PV^B for specifying the spatiotemporal and profile conditions, respectively. When a new moving object is being inserted into S^{STP}-tree, the first operation is to find a target leaf node that enlarges the $tpbr$ of the node smallest among all the leaf nodes. After inserting the object into the target leaf node, we update $tpbr$ and PV^B of the target leaf node if necessary. If $tpbr$ or PV^B of the parent node does not enclose all of its children as a result of inserting a new object into the target leaf node, we update them accordingly in the parent node. The same operation is applied to its parent node until the root node is reached recursively.

Relationships Between Authorization and Node: Given an authorization α and a node N, we are interested in different cases of spatiotemporal and PV^B relationships between α and N.

– *Spatiotemporal Relationship*
  - $\alpha^{\Box} \supset_{st} N^{\Box}$*:* spatiotemporal extent of α encloses that of N.
  - $\alpha^{\Box} \cap_{st} N^{\Box}$*:* spatiotemporal extent of α overlaps with that of N.
  - $\alpha^{\Box} \otimes_{st} N^{\Box}$*:* spatiotemporal extent of α is disjoint with that of N.

– *Profile Bounding Vector Relationship*
  - $\alpha^{\rightarrow} \supset_p N^{\rightarrow}$*:* $\alpha^{\rightarrow}$ encloses $N^{\rightarrow}$ if for each non-zero profile attribute vector[5] of $\alpha^{\rightarrow}$ and $N^{\rightarrow}$, bitwise 'OR' operation of $\alpha^{\rightarrow}$ and $N^{\rightarrow}$ results in $\alpha^{\rightarrow}$.
  - $\alpha^{\rightarrow} \cap_p N^{\rightarrow}$*:* $\alpha^{\rightarrow}$ overlaps with $N^{\rightarrow}$ if for each non-zero profile attribute of $\alpha^{\rightarrow}$ and $N^{\rightarrow}$, their bitwise 'AND' operation results in a non-zero profile attribute vector.
  - $\alpha^{\rightarrow} \otimes_p N^{\rightarrow}$*:* $\alpha^{\rightarrow}$ is disjoint with $N^{\rightarrow}$ if for each non-zero profile attribute of $\alpha^{\rightarrow}$ and $N^{\rightarrow}$, their bitwise 'XOR' operation results in all "1"s in the resultant vector.

Because the spatiotemporal relationships are straightforward, here we focus on profile bounding vector relationships between an authorization and a node. First, in case of $\supset_p$ relationship, observe that for every bit value of '0' of $\alpha^{\rightarrow}$, the corresponding bit value of $N^{\rightarrow}$ must be '0' because there must not exist any profile attribute value that only $N^{\rightarrow}$ includes but $\alpha^{\rightarrow}$ does not. Therefore, bitwise 'OR' operation would generate the same value with $\alpha^{\rightarrow}$. Also, in case of $\cap_p$ relationship, we need to see if there exists any common profile attribute value between α and $N^{\rightarrow}$. Therefore, if bitwise 'AND' operation results in a non-zero profile vector, we know that there exists common value set. Finally, in case of $\otimes_p$ relationship, we know α and $N^{\rightarrow}$ should not share

[5] A non-zero profile attribute vector refers to a binary vector that includes the value "1" in at least one bit.

Table 2. Bitwise Operation Results

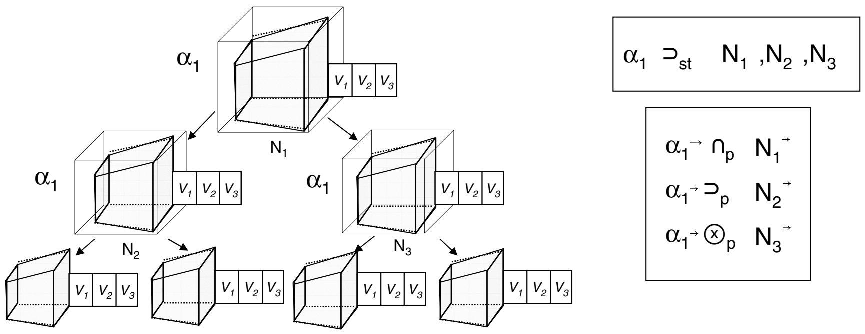
$\alpha^{\rightarrow}$	$N^{\rightarrow}$	AND	OR	XOR	Relationship
110	011	010	111	101	$\alpha^{\rightarrow} \cap_p N^{\rightarrow}$
110	010	010	110	100	$\alpha^{\rightarrow} \cap_p N^{\rightarrow}, \alpha^{\rightarrow} \supset_p N^{\rightarrow}$
110	001	000	111	111	$\alpha^{\rightarrow} \otimes_p N^{\rightarrow}$

any profile attribute value that is common to each other. The bitwise 'XOR' operation is used for checking this condition, and the result of 'XOR' must include all '1's in the resultant $N^{\rightarrow}$.

Suppose $\alpha^{\rightarrow} = \langle **, 110, ** \rangle$, which implies that the authorization α is given to the users with salary < \$62,000. Observe that because α evaluates the profile attributes 'Salary' only, we do not evaluate other profile attributes such as 'Department' or 'Home Town'. Therefore, as long as a user's salary is less than \$62,000, she meets the profile conditions of α. Considering the same PV_1^B, PV_2^B, PV_3^B in the previous section, suppose $N_1^{\rightarrow} = PV_1^B$, $N_2^{\rightarrow} = PV_2^B$, and $N_3^{\rightarrow} = PV_3^B$. We know that $N_1^{\rightarrow}$ and $N_2^{\rightarrow}$ include a user within this salary range while $N_3^{\rightarrow}$ does not include any user within the specified salary range. Also, in case of $N_2^{\rightarrow}$, all the value ranges of profile attributes for $N_2^{\rightarrow}$ are also included in $\alpha^{\rightarrow}$. Table 2 shows the results of bitwise AND, OR, XOR operations between $\alpha_1^{\rightarrow}$ and $N_1^{\rightarrow}$, $N_2^{\rightarrow}$, and $N_3^{\rightarrow}$ with their profile bounding vector relationships.

Authorizations Overlaying: The overlaying strategy traverses the S^{STP}-tree from the root node to leaf level by recursively comparing both the spatiotemporal extents and PV^Bs of the overlaying authorization and each node in the traversal path. Let us denote the spatiotemporal extent of a node N as $N^{\square}$, and PV^B as $N^{\rightarrow}$. All the possible scenarios for this comparison are as follows:

- **Case 1:** If $(\alpha^{\square} \supset_{st} N^{\square}) \wedge (\alpha^{\rightarrow} \supset_p N^{\rightarrow})$ is true, we stop traversing and overlay α on N. This overlaying strategy has several benefits. First of all, we overlay the authorizations on the first node encountered on the traversal path that totally encloses the spatiotemporal region and PV^B. As a result, authorizations are overlaid as high up as possible in the tree [4]. Because user access request evaluates also from the root node to the leaf level, authorizations that have been issued for the subject of the access request would be encountered as early as possible. Due to our overlaying strategy, existence of a relevant authorization in the traversal path for a subject of the access request means that all the moving objects stored at the subtree rooted at the node are already authorized. Therefore, we do not need to evaluate authorizations for the access evaluation process for the subtree. Observe that after overlaying an authorization on a node, it is not necessary to overlay the same authorization on any of its descendants.
- **Case 2:** Else if $(\alpha^{\square} \otimes_{st} N^{\square}) \vee (\alpha^{\rightarrow} \otimes_p N^{\rightarrow})$ is true, we stop the overlaying process. This is because, if subjects of α do not have a privilege to $N^{\square}$ or $N^{\rightarrow}$, α is not applicable to moving objects stored at the subtree rooted at N. Also, because $N^{\square}$ and $N^{\rightarrow}$ includes all the spatiotemporal extents and PV^Bs of all of N's descendants, there is no reason to traverse further to the leaf level.



α_1
V_1 V_2 V_3
N_1
N_2
N_3
$\alpha_1 \supset_{st}$ N_1, N_2, N_3
$\alpha_1^{\rightarrow} \cap_p$ $N_1^{\rightarrow}$
$\alpha_1^{\rightarrow} \supset_p$ $N_2^{\rightarrow}$
$\alpha_1^{\rightarrow} \otimes_p$ $N_3^{\rightarrow}$

Fig. 4. Authorization Overlaying Process in S^{STP}-tree

– **Case 3:** Else if $(\alpha^{\square} \cap_{st} N^{\square}) \vee (\alpha^{\rightarrow} \cap_p N^{\rightarrow})$ is true, the overlaying strategy is different depending on the level of N.
  - If N is a non-leaf node, we traverse to each of N's children node C, and the same comparison between α and C is processed. This is because there may exist a descendent node whose spatiotemporal extent and PV^B is enclosed by that of α.
  - If N is a leaf node, we overlay α on N. This is because at least one of the moving objects stored in N comply with the spatiotemporal and profile specification of α. Therefore, in order not to discard any relevant authorization, we need to overlay α on N.

Figure 4 presents the overlaying process in the S^{STP}-tree. It shows that a node N_1 is a root node of the tree, and N_2, N_3 are the children nodes of N_1. Consider an authorization α_1 to be overlaid on the S^{STP}-tree. α_1 cannot be overlaid on the node N_1 since $\alpha_1^{\rightarrow} \cap_p N_1^{\rightarrow}$, which belongs to the case 3 above. Therefore, we need to traverse down to N_1's children nodes N_2 and N_3. The first traversal path is to N_2, and α_1 can actually be overlaid on N_2 because $\alpha_1^{\square} \supset_{st} N_2^{\square}$ and $\alpha_1^{\rightarrow} \supset_p N_2^{\rightarrow}$, which is case 1. Another traversal path to N_3 is stopped because $\alpha_1^{\rightarrow} \otimes_p N_3^{\rightarrow}$, which belongs to case 2.

5 User Access Request Evaluation

In this section, we present the details of user access request evaluation. Typically, a user request is of the form of requesting objects in the area of interest that satisfy a certain profile criteria. For example, a merchant is interested in sending promotion deals to mobile customers who are near a mall and whose salary is greater than $52,000. However, such promotion deals should be reached to only to the customers who are willing to reveal their salary information to that merchant (specified in the authorization) to receive the promotion deal.

A user request is denoted as $U = \langle s, \square, V, m\rangle$ where s is the subject requesting access, $\square$ is the spatiotemporal extent that the subject is interested in, V is interested profile vector, and m the access mode. We denote $U.s$, $U^{\square}$, $U^{\rightarrow}$, and $U.m$ to denote the subject, the spatiotemporal extent, the profile vector, and the access mode of the user access request U, respectively. For example, if a merchant A wants to locate mobile customers who are 10 miles from the shopping mall and whose salary is greater than \$52,000, the user request would be U =⟨merchant_A, circle((50,60),10),⟨011⟩, locate⟩.

Algorithm 1. UserAccessRequestEvaluation

```
1: **Input:** node $N$, User Access Request $U$, Boolean $authorized$
2: **Output:** a set of authorized moving objects $resultSet$
3: if $U^\square \otimes_{\{x,y,t\}} N^\square$ OR CheckPV($U^\rightarrow, N^\rightarrow$) $= disjoint$ then
4:     return $NIL$
5: end if
6: if There exists overlaid authorizations in $N$ then
7:     $\Lambda(N) \leftarrow$ CheckUserIDAuth($U, N$)
8: end if
9: $resultSet \leftarrow NIL$
10: if $authorized = false$ AND $\Lambda(N) \neq \emptyset$ AND $N$ is a non-leaf node then
11:     if $\Lambda(N) \neq NIL$ AND $U^\square \cap_{\{S,T\}} N^\square$ AND CheckPV($U^\rightarrow, N^\rightarrow$) $= disjoint$ then
12:         $authorized \leftarrow true$
13:     end if
14: else if $authorized = false$ AND $\Lambda(N) \neq \emptyset$ AND $N$ is a leaf node then
15:     for each $\alpha$ in $\Lambda(N)$ do
16:         if $N^\square \cap U^\square$ AND CheckPV($U^\rightarrow, N^\rightarrow$) $= overlap$) then
17:             $resultSet \leftarrow resultSet \cup$ evaluate($\alpha, U, N$)
18:         end if
19:     end for
20:     return $resultSet$
21: end if
22: if $authorized = true$ AND $N$ is a leaf node then
23:     return evaluate($U, N$)
24: end if
25: for each child $c$ in $N$ do
26:     MUserAccessRequestEvaluation($c$, $U$, $authorized$)
27: end for
```

Algorithm 1 discusses the details of user access request processing. The initial function call is UserAccessRequestEvaluation($R, U, false$) where R is a root node of S^{STP}-tree. The evaluation process starts with the root node by comparing the spatiotemporal extents and the profile vectors of the user request and each node N involved in the top-down traversal. At the same time, the evaluation process searches for the relevant authorizations.

Given a user request U, we say an authorization α as relevant to U if the set of subjects evaluated by $\alpha.se$ includes $U.s$ and $U.m \prec_m \alpha.m$ for $U.s$ overlaid on the node N. We denote the relevant authorizations at a node N on the tree as $\Lambda(N)$ ={$\alpha \in$ overlaid authorizations on $N \mid U.s \in \alpha.se$, $U.m \prec_m \alpha.m$ }. The comparison among U, N and $\Lambda(N)$ during the traversal results in the following cases.

Case 1: $(U^{\Box} \otimes_{st} N^{\Box}) \vee (U^{\rightarrow} \otimes_{p} N^{\rightarrow})$ is true: The disjoint relationship implies that all the moving objects stored at the subtree rooted at N are not within the spatiotemporal region or do not meet the profile condition for the user request U. Regardless of the existence of relevant authorizations for U at N, the moving objects stored at the subtree rooted at N are not within the user's interests. Therefore, the traversal stops regardless of the existence of overlaid authorizations.

Case 2: $(A(N) \neq \emptyset) \wedge ((U^{\Box} \cap_{st} N^{\Box}) \vee (U^{\rightarrow} \cap_{p} N^{\rightarrow}))$ is true: If N is a non-leaf node, although all the moving objects stored at the subtree rooted at N are authorized, the user wants to retrieve a subset of moving objects whose locations are within $U^{\Box}$ and whose profiles are enclosed by $U^{\rightarrow}$. Therefore, for the subtree rooted at N, we retrieve moving objects whose location overlaps with $U^{\Box}$ and whose profile condition overlaps with $U^{\rightarrow}$. We do not need to evaluate authorizations during the traversal because the subtree rooted at N is already authorized by $A(N)$.

If N is a leaf node, because we overlay authorizations on a leaf-node in an enclosing case as well as overlapping case, not all of the moving objects in N are authorized. Thus, for all $\alpha \in A(N)$, return the moving objects that are located within $\alpha^{\Box} \cap_{st} U^{\Box}$ and whose profiles are overlapped with $\alpha^{\rightarrow} \cap_{p} U^{\rightarrow}$.

Case 3: $(A(N) = \emptyset) \wedge ((U^{\Box} \cap_{st} N^{\Box}) \vee (U^{\rightarrow} \cap_{p} N^{\rightarrow}))$ is true: If N is a non-leaf node, access control decision cannot be made because there is a possibility that a relevant authorization may be overlaid on a descendent node of N. Thus, evaluation process repeats for all the children nodes of N. If N is a leaf node, we reject the access request because there exists no relevant authorization for U during the traversal.

Case 4: $(A(N) \neq \emptyset) \wedge ((\mathrm{U}^{\Box} \supset_{st} N^{\Box}) \wedge (U^{\rightarrow} \supset_{p} N^{\rightarrow}))$ is true: There exists at least one relevant authorization for U is overlaid on N. If N is a non-leaf node, because the spatiotemporal extents and profiles stored at the subtree rooted at N are authorized, all the moving objects stored at leaf nodes of the subtree rooted at N are allowed to be accessed by $U.s$. Therefore, there is no need to evaluate authorizations on the subtree rooted at N. In addition, spatiotemporal and profile vector comparisons would not be required either because all the moving objects stored at the subtree rooted at N are within the user's interests. If N is a leaf node, some of the moving objects in N may not meet the conditions set by U. Thus, for all $\alpha \in A(N)$, the algorithm would return all the moving objects that are located within $\alpha^{\Box}$ and whose profiles are overlapping with those of $\alpha^{\rightarrow}$.

Case 5: $A(N) = \emptyset \wedge ((U^{\Box} \supset_{st} N^{\Box}) \wedge (U^{\rightarrow} \supset_{p} N^{\rightarrow}))$ is true: Although all the moving objects stored at the subtree rooted at non-leaf node N meet the spatiotemporal and profile conditions of U, access control decision cannot be made because there is a possibility that a relevant authorization may be overlaid on a descendent node of N. Thus, evaluation process repeats for all the children nodes of N. If N is a leaf node, we reject the access request because there exists no relevant authorization for U.

Table 3. Number of Disk Access

S^{STP}-Tree		Separate Index Structures	
If $m < M - k/s$	Else	Index for moving objects	Index for profiles
$\Omega(\log_m N)$	$\Omega(\log_{M-k/s} N)$	$\Omega(\log_m N)$	$\Omega(\log_m N)$

Note that, in algorithm 1, the operation CheckUserIDAuth() returns relevant authorizations for U among the overlaid authorizations in the node N. CheckPV(A, B) returns *overlap* (if $A \cup_p B$), *disjoint* (if $A \otimes_p B$), and *enclose* (if $A \supset_p B$). The overloading function evaluate() returns the moving objects whose location and profile conditions meet the user request, and which are stored in the leaf node N.

User Access Request Performance Analysis: We present an informal analysis of the complexity of user request evaluation by comparing the performance between the proposed S^{STP}-tree and the where there are two separate index structures (one for moving objects and another for profile). For the discussion, we do not consider authorizations because the overlaying procedure does not change the structure of the tree. Overlaying simply stores the relevant authorizations on the nodes of the tree, which does not incur any changes on the structure of the tree.

For the analysis, let us suppose the following:

– N is the number of moving objects: That is, there are N number of location information and N of profile vectors.
– The number of children (or data) that each non-leaf (or leaf) node includes is between m and M where 2 $\leq$ m $\leq$ M/2: this implies that the height of the tree is bounded by [$\log_M N$, $\log_m N$]
– k is the size of PV^B in bytes
– b is the disk page size in bytes
– s is the size of location information and profile vector (in bytes)

If a tree does not store a PV^B in a node, $M = b/s$ because M is the maximum number of children node or data that can be stored in each disk page without considering the PV^B. However, in case of S^{STP}-Tree, k bytes are reserved to store a PV^B for each node. Thus, the maximum number of data (children) for each disk page is

$$(b - k)/s = b/s - k/s \quad (1)$$
$$= M - k/s \quad (2)$$

Thus, the height of S^{STP}-tree is
– If $m < M - k/s$, the height = [$\log_m N$, $\log_{M-k/s} N$].
– Else, the height = [$\log_{M-k/s} N$, $\log_m N$]

The number of disk accesses for user request is summarized in 3. If $m < M - k/s$, it is obvious that the S^{STP}-tree shows better performance (fewer disk accesses) because in this case, the proposed tree would generate the exactly same structure of tree as that from the separate index for moving objects. Thus, separate indexes would need to access the number of nodes from the profile index additionally than the S^{STP}-tree.

6 Conclusions

Recently, unified index structures, ^{s}TPR-tree and S^{PPF}-tree, have been proposed to organize both moving objects and authorizations specified over them. However, both approaches are not capable of evaluating security policies that include profiles because the index structure can support only spatiotemporal regions of security policies. In this paper, we have proposed an index structure, called the S^{STP}-tree, which enables one to enforce and evaluate the security policies that include profiles of moving objects. We provide an informal analysis to show that the proposed structure is more efficient than maintaining indexes independently for moving objects and authorizations, and profiles. Currently, we are in the process of implementing the proposed tree to experimentally validate the gain in performance with respect to the independent cases.

References

1. Youssef, M., Adam, N.R., Atluri, V.: Preserving Mobile Customer Privacy: An Access Control System for Moving Objects and Customer Information. In: 6th International Conference on Mobile Data Management. LNCS. Springer, Heidelberg (2005)
2. Wolfson, O., Xu, B., Chamberlain, S., Jiang, L.: Moving objects databases: Issues and solutions. In: Rafanelli, M., Jarke, M. (eds.) 10th International Conference on Scientic and Statistical Database Management, Proceedings, Capri, Italy, July 1-3, 1998, pp. 111–122. IEEE Computer Society Press, Los Alamitos (1998)
3. Moreira, J., Ribeiro, C., Abdessalem, T.: Query operations for moving objects database systems. In: Proceedings of the eighth ACM international symposium on Advances in geographic information systems, pp. 108–114. ACM Press, New York (2000)
4. Atluri, V., Guo, Q.: Unied 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)
5. Atluri, V., Shin, H.: Efficient enforcement of security polices based on the tracking of mobile users. In: DBSec, pp. 237–251 (2006)
6. Saltenis, S., Jensen, C.S., Leutenegger, S.T., Lopez, M.A.: Indexing the positions of continuously moving objects. In: SIGMOD Conference, pp. 331–342 (2000)
7. Atluri, V., Adam, N.R., Youssef, M.: Towards a unied index scheme for mobile data and customer proles in a location-based service environment. In: Workshop on Next Generation Geospatial Information (NG2I'03) (2003)
8. Atluri, V., Chun, S.A.: An authorization model for geospatial data. IEEE Trans. Dependable Sec. Comput. 1(4), 238–254 (2004)
9. Gazetteer, U.S.: http://www.census.gov/cgi-bin/gazetteer
10. Oracle corporation data sheet - oracle spatial option and oracle locator: Location features in oracle database 10g. Technical report, Oracle (2004)http://www.oracle.com/
11. Atluri, V.: Mobile commerce. In: In The Handbook of Computer Networks, Volume III Distributed Networks, Network Planning, Control, Management and Applications, Part 3: Computer Network Popular Applications, John Wiley & Sons Inc., West Sussex, England (2007) (page to appear)

12. Mobile Commerce (M-Commerce) & Micropayment Strategies. Technical report, Juniper Research (2004) http://www.juniperresearch.com/
13. Venkatesh, V., Ramesh, V., Massey, A.P.: Understanding usability in mobile commerce. Commun. ACM 46(12), 53–56 (2003)
14. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: International Conference on Machine Learning, pp. 194–202 (1995)

Reliable Delivery of Event Data from Sensors to Actuators in Pervasive Computing Environments[⋆]

Sudip Chakraborty, Nayot Poolsappasit, and Indrajit Ray

Computer Science Department
Colorado State University
Fort Collins, CO 80523, USA
{sudip, nayot, indrajit}@cs.colostate.edu

Abstract. The event-condition-action (ECA) paradigm holds enormous potential in pervasive computing environments. However, the problem of reliable delivery of event data, generated by low capability sensor devices, to more capable processing points and vice versa, needs to be addressed for the success of the ECA paradigm in this environment. The problem becomes interesting because strong cryptographic techniques for achieving integrity impose unacceptable overhead in many pervasive computing environments. We address this problem by sending the data over the path from the sensor node to the processing point that provides the best opportunity of reliable delivery among competing paths. This allows using much weaker cryptographic techniques for achieving security. The problem is modeled as a problem of determining the most reliable path – similar to routing problems in networks. We propose a trust-based metric for measuring reliability of paths. The higher the trust value of a path the more reliable it is considered. We propose techniques for estimating the trust levels of paths and propose a new algorithm for identifying the desired path.

1 Introduction

Pervasive computing technology has the potential to impact numerous applications that benefit society. Examples of such applications are emergency response, automated monitoring of health data for assisted living, environmental disaster mitigation and supply chain management. Pervasive computing uses numerous, casually accessible, often invisible, computing and sensor devices. These devices are frequently mobile and/or embedded in an environment that is mobile. Most of the time they are inter-connected with each other, with wireless or wired technology. Being embedded in the environment and interconnected allow pervasive computing devices to exploit knowledge about the operating environment in a net-centric manner. This enables pervasive computing applications to provide a rich new set of services and functionalities that are not otherwise possible through conventional means. Pervasive computing applications frequently rely

⋆ This work was partially supported by the U.S. Air Force Research Laboratory (AFRL) and the Federal Aviation Administration (FAA) under contract F30602-03-1-0101 and by the U.S. Air Force Office of Scientific Research under contract FA9550-07-1-0042. Any opinions, findings, and conclusions expressed in this publication are solely those of authors and do not necessarily represent those of the AFRL, the FAA, or the AFOSR.

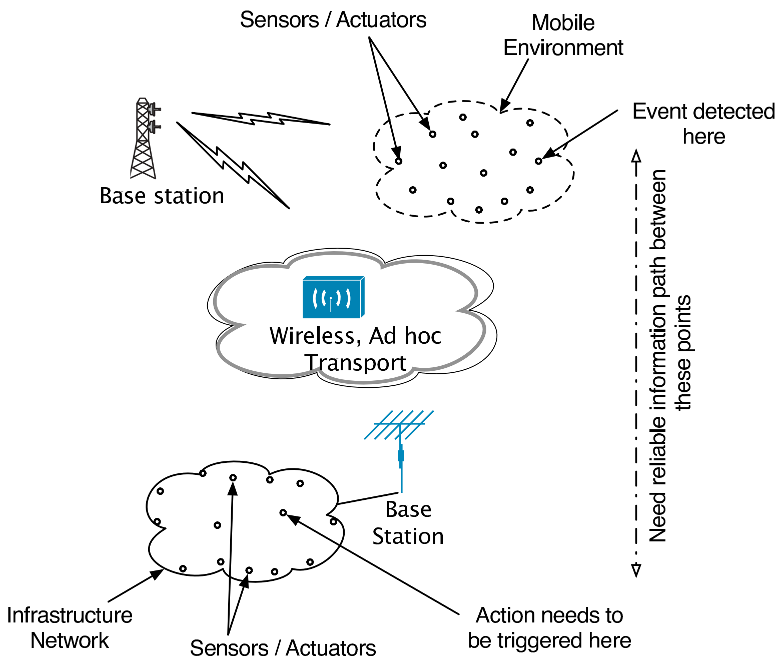


Fig. 1. Pervasive computing environment involving remote event detection and action triggering

on event-triggered obligation policies to operate in a dynamic environment. An obligation policy is associated with events, conditions, subjects, objects and actions. When the event of interest occurs and the associated conditions evaluate to true, the subjects perform the specified actions on the objects. Events are typically identified and captured by embedded sensing devices and actions are actuated by similar embedded devices. Processing of captured event data for evaluation of conditions are, on the other hand, mostly performed at remote processing nodes or base stations. This is because the sensing and actuating devices embedded in the environment are frequently of very low performance capabilities including low computing, low storage and low power. Thus a major challenge in a pervasive computing environment is to provide a path for propagating sensor data to processing nodes and action data to actuating nodes. Reliability of the paths is important. The data should be delivered with the minimum possible error and in as timely a manner as possible.

The reliable transmission path requirement imposes significant challenges in pervasive computing environments. A pervasive computing application can seldom assume a reliable network infrastructure for communication. In a conventional setting, a node that generates a message forwards it to a neighboring reliable node. The receiving node in turn forwards the message to another fixed node that is known a priori. This procedure is followed till the message reaches the destination. Every node in this process knows at least one other reliable node in the path towards the destination to which the message can be handed over. Frequently a node will know about more than one other node and thus have a choice of a better node. The nodes are static, that is they do not change their location and consequently the links between the nodes are stable. This and the proper use of strong cryptographic techniques, easily facilitate reliable delivery of messages in conventional settings. In a pervasive computing environments, on the other hand, mobility of nodes (sensing, processing or actuating) is frequently considered an asset. Figure 1 depicts the scope of the problem. Nodes are not locationally stable; instead they continuously change their coordinates. Thus, a node that needs a message delivered cannot rely on another fixed node to forward the message but has to make use of one or more nodes that happen to be within reachable distance at that particular moment. In addition, since a majority of these nodes are low capability devices (in the sense of low computational capabilities, low storage and low power provisions), use of strong cryptographic techniques needs to be ruled out. Moreover, in hostile environments these nodes get easily compromised. Under such circumstances it will enormously benefit a pervasive computing application if the path that provides most opportunity of reliable delivery of messages is presented to it. Determining an appropriate path within a network is the problem of routing. In this paper we revisit this problem in the context of pervasive computing environments.

The problem of routing in mobile ad hoc networks have been addressed before [1,2,3,4,5,6,7,8,9,10,11]. Among these [1,6,7,10] study cryptographic techniques for securing the routing protocol. Some use public key cryptography to encrypt the end-to-end transmission of routing messages. Others use digital signature techniques to authenticate routing messages at the peer-to-peer level connection. However, these cryptographic techniques incur high computation and storage overhead which limit their use in sensor devices. Use of secret key techniques instead of public keys alleviate this problem to some extent although at the expense of added complexity. Moreover, key distribution and management is a big problem in secret key based systems. It is difficult to establish a key distribution or certification authority in mobile ad hoc environments. Ensuring the availability of a key distribution center or a certifying authority is almost impossible given the unstable nature of the network.

The Hermes protocol developed by Zouridaki et al. [11] proposes using trustworthiness of its neighbors for routing. The trust values are computed under the assumption that they follow the beta probability distribution. The parameters of the beta distribution come from the empirical observation of the forwarding behavior. Thus, nodes that maintain a good and steady forwarding history have more trust and confidence on them. The route is established for the most trusted path. However, the major problem of this work is its complete reliance on forwarding history for measuring trust. A malicious node can easily fake this history thus presenting itself as a trusted node. Other similar works include [12,13,14]. Among these [13] proposes a signal stability-based adaptive routing (SSR) where the routes are selected based on signal strength. This work looks promising; however it does not discuss how to measure this signal strength quantitatively. In [12] the authors propose an on demand secure routing protocol where the metric is based on past history. Yi et al. [14] present a security-aware ad hoc routing protocol (SAR) in which a route is selected on the basis of degree of 'security guarantee' that the route provides. If two routes have same guarantee then the shorter path is chosen.

The security metric can be specified by standard security properties like timeliness, ordering, authenticity etc. However, the paper does not discuss how we can measure these properties quantitatively.

We propose a trust-based routing protocol for pervasive computing environments. Our protocol determines the most reliable path under currently determinable properties of the system to forward a packet from source to destination. A node in the pervasive computing environment is any entity that is able to forward a packet. It can be a sensor node, a mobile device like a PDA or a cellular phone, a powerful computing device or even an actuator like a switch. Reliability of a node is measured in terms of a *trust* value for the node. Each node determines its neighbor's trust based on physical properties of the neighbor that can be directly observed, the neighboring node's behavior history (i.e., results of past interactions) and recommendation (or rating) about the neighbor from other neighbors. The resulting trust value is used to generate the 'cost of forwarding', or simply *cost*. The cost metric is inversely related to the trust metric, that is, higher the trust (reliability) on the node, lower is the cost. This cost is associated with links in the network. We next modify the widely used distance vector routing protocol using these costs between the links to find the path with minimum average cost. The chosen path then becomes the most 'reliable' path.

The rest of the paper is organized as follows. Section 2 gives an overview of our protocol including a discussion on cost function in subsection 2.1. We introduce our trust metric in section 3 where the components and the methods to compute them have been discussed in subsections 3.1, 3.2, and 3.3 followed by the computation of the final trust value in subsection 3.4. Subsequent section 4 presents our trust-based routing protocol. In section 5 we analyze our protocol. We present the security analysis of the protocol in subsection 5.1. In subsection 5.2 we discuss the complexity of our protocol. We start with computation complexity followed by communication complexity and storage complexity respectively. Finally, we conclude our discussion in section 6 with a summary for future work.

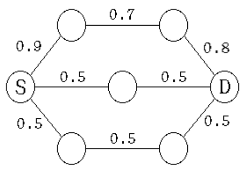
2 Overview of Trust-Based Routing Protocol

We assume that the pervasive computing environment supporting the application has a very dynamic topology. Nodes join or depart the environment at random. Each node in the network maintains a table RT consisting of tuples of the form $\langle Dest, Win\#, NH, Cost_{avg}, \sum Cost^2, \#Hops\rangle$. In this table the node stores on a per-destination ($Dest$) basis, the identity of the next neighbor (NH), to which the message needs to be forwarded. Together with the next neighbor information, the node also stores the minimum average cost ($Cost_{avg}$) and the number of hops ($\#Hops$) to reach the particular destination. This information is generated periodically. Thus each tuple bears a time stamp in the form of a current time window ($Win\#$). The routing algorithm that we propose is used to update the next neighbor entry in the RT table for a particular destination.

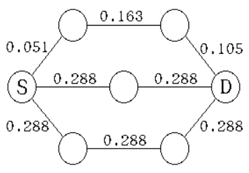
A source node initiates the routing protocol if it has a packet to be sent to a destination for which it does not have any next hop (NH) entry. The source node can also execute the routing protocol when the path to the destination has expired (when the value under $Win\#$ is less than the current window number). The source sends out a route discovery request. We assume that each node that participates in the pervasive computing environment has a *trust relationship* with its neighbor (that is a node at 1 hop distance). A trust-aware node periodically sends a *beacon* message to its neighbors. The beacon message is something like an "I am alive" message and carries information necessary to prove the node's existence. Once in a while a node can also send out a beacon message on demand. In our protocol, a node may request a *recommendation* score from a second node about a neighbor of the second node. In such cases the recommendation score is carried on a beacon message. Beacon messages are broadcast in nature. They carry rudimentary checksums to provide weak protection against integrity violations. The recommendation score is used as one of the parameters for computing *trustworthiness* of a neighbor. Trust relationships are periodically refreshed locally. At some point after system initiation we assume that every node in the system will have a trust relationship with each of its neighboring nodes. We do not assume that trust relationships are symmetric or transitive.

We adapt the trust model proposed in [15]. We express the trust relationships between nodes as $N_r \longrightarrow N_e$ where N_r is the *truster* node and N_e is the *trustee* node. We represent this trust relationship as a tuple $({}_{N_r}P_{N_e},\ {}_{N_r}R_{N_e},\ {}_{N_r}I_{N_e})$. The value ${}_{N_r}P_{N_e}$ represents an evaluation of the *physical properties* of N_e by N_r. The value ${}_{N_r}R_{N_e}$ denotes an evaluation of the *recommendation scores* of N_e from other nodes and ${}_{N_r}I_{N_e}$ evaluates the *interactions* that N_r had with N_e. The exact interpretation of these terms are deferred for the time being till the next section (section 3). We associate a numeric value $v(N_r \longrightarrow N_e)$ (from $[-1,1]$) with the above tuple which we refer to as the trust value for node N_r on node N_e along the edge $(N_r,\ N_e)$. We next convert this trust value to a cost on the link $(N_r,\ N_e)$. The higher the trust value the lesser is the cost to transfer messages on the link. The path having the least average cost from the source node to the destination node is considered the most reliable among the available paths and is chosen by the source node to forward the data.

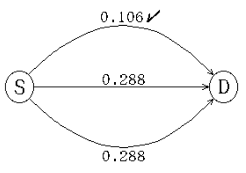
Figure 2(a) describes pictorially the main idea of our protocol. We assume that if a node N_r has a distrust value (that is value less than 0) on another node N_e the cost on that link is infinite and that next hop is discarded. When a node receives a route discovery request from a source, it checks its routing table RT. If a route to the destination is present in RT which has not expired, it sends the '#$Hops$' and cost related information



(a) The trust relations in trust-aware pervasive computing environment



(b) The forwarding cost in trust-aware pervasive computing environment



(c) Path having least average cost is most reliable

Fig. 2. Trust relation between nodes and the corresponding cost on the link

to the source. The source then evaluates the cost of the link between the neighbor and itself and using the $\#Hops$ it computes the average cost of forwarding the packet to the destination. The source may get multiple such responses. It then chooses the next hop for which the average cost over the path is minimum. If the node that receives a route discovery request from the source, does not itself have the next hop information in its RT for that destination, it initiates a route discovery process as a source. This process can go on till the node which is 1 hop behind the destination initiates a route discovery request.

2.1 Cost Function

Each node tries to find the path to a given destination which has the minimum average forwarding cost. The *cost* of forwarding a packet from the node N_r to N_e is a function of $v(N_r \longrightarrow N_e)$. These two are related as follows: higher the trust, lesser is the cost and the cost increases as the distrust increases (i.e., the trust decreases). Rationale is, the cost (in terms of integrity violation and other malicious activities) of forwarding a packet through a more trustworthy node is less than that through a less trustworthy node. The cost is minimum (not zero though) when N_r has absolute trust ($v(N_r \longrightarrow N_e) = 1$) on N_e. This minimum cost (Min_{cost}) is a small positive cost incurred due to forwarding overhead. It is uniform over the whole pervasive computing environment and set at the bootstrapping of the system. We assume that the decay in cost with increased trustworthiness is logarithmic with the following conditions: at $v(N_r \longrightarrow N_e) = 1$, cost $= Min_{cost}$ and at $v(N_r \longrightarrow N_e) = -1$, cost $= \infty$. The function is defined as,

$$cost(N_r,\, N_e) = Min_{cost} - \ln(\frac{1 + v(N_r \longrightarrow N_e)}{2}) \tag{1}$$

The maximum allowable cost for N_r is incurred when $v(N_r \longrightarrow N_e) = 0$. This corresponds to the situation when N_r is *neutral* about trustworthiness of N_e. This cost, denoted by $MaxAllowed_{cost(N_r,\, N_e)}$, is $Min_{cost} - \ln(\frac{1}{2})$, that is, $MaxAllowed_{cost(N_r,\, N_e)} = Min_{cost} + 0.69$.

Computing this cost value has some overhead but is only linear in the number of nodes in the pervasive computing environment. The cost value is stored in the mobile device for a predetermined time or until a new beacon message has arrived. The absence of a beacon message from a particular node in a particular window of time represents a broken link during that time period. It can happen for various reasons including that the node is compromised. The node in such a case may either discard the broken link from the list of current neighbors or mark it as unused. Routing information is advertised by broadcasting the route setup packets periodically or on demand depending on the protocol used. These packets indicate which mobile nodes are accessible from which others and the average cost associated with the path towards a destination. When a node receives a data packet, it chooses the path which has the lowest average forwarding cost and forwards the packet to the neighbor on this path to be further forwarded towards the destination. During this process, a node also evaluates the packet forwarding performance of the neighbor node. By measuring this the node essentially evaluates an interaction score for the neighbor. The details of this process and other routing processes is explained in section 4 and section 5.2.

3 Trust Metric

As mentioned earlier in section 2 the trust of N_r on N_e depends on three factors – N_e's *properties*, *recommendation* about N_e (alternatively, N_e's rating) from another node N_k, and N_r's *interaction* with N_e. We assume that each of these three factors is expressed in terms of a numeric value in the range $[-1,1]$. A negative value for the component is used to indicate the *trust-negative* type for the component, whereas a positive value for the component is used to indicate the *trust-positive* type of the component. A 0 (zero) value for the component indicates *trust-neutral*. The final trust value, denoted by $v(N_r \longrightarrow N_e)$, is calculated as an average of these component values. Eventually $v(N_r \longrightarrow N_e)$ falls in the range $[-1,1]$. A trustee node is completely trusted (or distrusted) if the value of the trust relationship is 1 (-1). If the value is in the range $(0,1)$ the node is *semi-trustworthy*; if the value is in the range $(-1,0)$ the node is *semi-untrustworthy*. The 0 value represents trust neutrality, that is the trustee is equally trustworthy as untrustworthy.

3.1 Computing *properties*

In our approach trust is used as a reliability metric of a neighbor node for proper handling and forwarding the packet to the destination. A node N_i is a neighbor of node N_j if N_i is within the range of a beacon message from N_j. A node becomes more reliable when it has relatively more resources (in terms of signal strength, signal stability, less propensity to corrupt data etc.). Higher values of these attributes show that it is more capable of handling and forwarding a packet in a reliable manner. This motivates us to measure the node properties quantitatively and include that measure as a factor to evaluate trustworthiness of a node. We focus on two properties of a node – signal strength, and stability factor. A node maintains a property table, $PT = \langle Node\_id, SS_{avg}, SF \rangle$ for each neighbor node where the properties of the neighbor is kept along with the corresponding id. The table is updated after each time-window *win*.

Measuring signal strength. In each time-window *win*, a node periodically sends a link layer beacon message to its neighbors. When the neighbor node receives such a beacon message, the extended device driver interface of the receiving node measures the signal strength at which the beacon was received. In our approach we use the *receive signal strength indicator* (RSSI) unit to measure the signal strength. RSSI is the IEEE 802.11 standard for measuring radio frequency (RF) energy sent by the circuitry on a wireless network interface card (W-NIC). It is a numeric integer value with an allowable range of 0 to 255. However, for the sake of our model we give a transformation to this recorded signal strength value by dividing it by 255. This scales the received signal strength value within the range $[0,1]$. We require this transformation as the final value of the component 'properties' lies within $[-1,1]$. At the end of each time-window, we take the average of these values. This transformed average signal strength value is then stored under the column SS_{avg} in the table PT corresponding to the neighbor. All these signal strength values within the time-window *win* is kept in a separate temporary property table, $PT_{tmp}^{Node\_id} = \langle SS, SB \rangle$ where SB is the *stability bit* explained next.

Measuring stability factor. The stability factor indicates the stability of a node. Higher the stability more reliable is the node to forward a packet. We derive the stability factor using signal strength. The reason is as follows: if a node is locationally unstable, it will have a varying signal strength. Alternatively, if the average signal strength of a node is fairly constant over few time-windows, the link with the node can be considered as stable. Therefore, after storing the strength of received signal, say $SS_{current}$, in $PT_{tmp}^{Node\_id}$, it is compared with the value present in SS_{avg} of PT. If $SS_{current} < SS_{avg}$ then the SB is set to 0, otherwise the default value 1 is kept. At the end of time-window, the stability factor SF is calculated as,

$$SF = \frac{\text{number of bits set to 1 under SB}}{\text{Total number of bits under SB}}$$

At the beginning of each time window the temporary property table $PT_{tmp}^{Node\_id}$ is set to its default values. The default value for SS is 0 and for SB is 1.

Measuring properties. The *properties* component of the node N_e is then computed as

$$_{N_r}P_{N_e} = \alpha * SS_{avg} + (1-\alpha) * SF \tag{2}$$

where $\alpha \in (0,1)$ is a fraction used as the relative importance weight to the signal strength property.

3.2 Computing *recommendation*

Each trust-aware node agrees to provide a 'recommendation' about its neighbors upon receiving a *recommendation request* from a source node. Let N_r request N_k for a recommendation about a node N_e. The source node N_r sends this recommendation request by sending a special message REC_REQ containing the node_id of the target node (in this case N_e). The node N_r can choose this recommender using a *threshold_trust* value T_{thr}. That is, if $v(N_r \longrightarrow N_k) \geq T_{thr}$ then only N_r sends a REC_REQ message to N_k. If N_k has a trust relationship with N_e, then N_k replies by sending a message REC_RESPONSE containing the pair $\langle node\_id, V\rangle$ where $V = v(N_k \longrightarrow N_e)$. The node N_r then scales this recommendation with the trust value that it has on N_k. Averaging all such recommendation received gives the 'recommendation' (or, rating) of N_e. Formally, if N_r receives m recommendations about N_e, then

$$_{N_r}R_{N_e} = \frac{1}{m}\sum_{k=1}^{m}\{v(N_r \longrightarrow N_k) \times v(N_k \longrightarrow N_e)\} \tag{3}$$

The truster N_r maintains a list, called *recommendation list*, $RL = (node\_id, list)$ for each trustee where structure of each item in the list is (node_id, recommendation_value).

3.3 Computing *interaction*

Interaction is modeled as cumulative effect of events encountered by a truster node N_r regarding N_e. We classify interaction in two categories – *packet forwarding interaction* – when the truster considers the behavior of the trustee as a packet forwarder, and *rating* *interaction* – when the truster considers the behavior of the trustee as a recommender. Every event in each of these categories has binary outcome; either the truster N_r has trust-positive event or a trust-negative event depending whether the event contributes toward a trust-positive interaction or a trust-negative interaction.

Evaluating packet forwarding interaction. To evaluate packet forwarding interaction, the truster N_r checks the outcome of each packet forwarded to N_e within the specific time window *win*. Each packet forwarded correctly towards the destination is considered as a trust-positive event. Each dropped packet gives rise to a trust-negative event. The node N_r measures the number of forwarded packet by N_e as follows: N_r forwards packets to N_e and with every such packet N_r sends an ECHO message with a *time-to-live* (TTL) $= 2$. Each reply received by N_r denotes correct forwarding of the packet by N_e to the next member N_k in the path. N_r keeps this information in a table $IT = \langle Node\_id, PFC_p, PFC_n, RC_p, RC_n \rangle$ where PFC_p denotes the counter for trust-positive packet forwarding interaction within the window and PFC_n counts the trust-negative packet forwarding interactions. RC_p and RC_n are rating counters used for counting the results of rating interactions. All these fields has default value 0. Whenever a packet is dropped the counter PFC_n is increased by 1 and for each received reply the counter PFC_p is increased. Formally, *packet forwarding interaction*, denoted by I_{pf} of N_r about N_e within the window *win* is defined as the ratio $\frac{PFC_p - PFC_n}{PFC_p + PFC_n}$.

Evaluating rating interaction. The *rating interaction* is evaluated in a similar manner. We assume that each node agrees to provide a 'trust recommendation' about its neighbors upon receiving a *recommendation request* from a source node. We also assume that for each neighbor, the truster keeps a list of node_ids of the nodes who have provided recommendation for that neighbor. Whenever the truster has a 'packet forwarding interaction' with the neighbor node and the result of that interaction matches with the recommendation, that is the truster has positive (negative) experience and the recommendation is also positive (negative), it increases the RC_p in IT of all such recommenders by 1. If there is a mismatch between the outcome and the recommendation, the truster increases the RC_n counter by 1 for those recommenders. For example, let the trustee N_e has provided "positive" recommendations for the nodes N_i, N_j, N_k to the truster N_r. Therefore, in the recommender list RL, N_e appears in the list against each of these nodes. Let N_r have trust-positive packet forwarding interaction with N_i, N_j and trust-negative packet forwarding interaction N_k. Then in the interaction table IT, for the node N_e, the counter RC_p is increased twice and RC_n once. At the end of time window *win* the *rating interaction*, denoted by I_r of N_r about N_e is defined as the ratio $\frac{RC_p - RC_n}{RC_p + RC_n}$.

Evaluating interaction. The *interaction* component of the node N_e is evaluated as

$$_{N_r}I_{N_e} = \beta * I_{pf} + (1 - \beta) * I_r \tag{4}$$

where $\beta \in (0, 1)$ is a fraction used as the relative importance weight to the packet forwarding interaction.

3.4 Computation of Final Trust Value

After computing values of the components we evaluate the trust value for the trustee N_e as the average of the components. Formally,

$$v(N_r \longrightarrow N_e) = \frac{{}_{N_r}P_{N_e} + {}_{N_r}R_{N_e} + {}_{N_r}I_{N_e}}{3} \tag{5}$$

These information are kept in a trust table, $TT = \langle$node_id, properties, recommendation, interaction, trust_value, cost$\rangle$. After each window *win* this table is updated with new values which are kept and used in the next time window. All other tables are set to their corresponding default values. Next section describes the modified distance vector routing algorithm which finds the path with minimum average cost for forwarding a packet to the destination.

4 Data Path Discovery

To select the most trustworthy path, each node evaluates and dynamically updates the trust components between itself and current neighbors. It then calculates the trust value of the neighbors by the process described in section 3. These values are used to calculate the forwarding cost between two neighbors, using the equation 1 in section 2.1. The path with the minimum average forwarding cost is preferred and the adjacent node on this path is trusted to forward the packets toward the destination.

4.1 Route Discovery

Our algorithm is based on a "rumor" about paths from neighbors. This is incomplete information. We thus choose to use the average and standard deviation of the running sum of cost in our route discovery protocol. This formula does not require the complete path information yet can correctly evaluate the path's reliability like the one with the complete path information. The average and standard deviation of running sum are computed as follows: let, a random variable X take on the values $x_1, \ldots, x_n$ and x be the latest value. We use the following equations:

$$AVG = (x + \sum_{i=1}^{n} x_i)/(n+1) \tag{6}$$

$$SD = \sqrt{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2/n(n-1)} \tag{7}$$

When a node receives a route information message from a neighbor node N_k, it updates the forwarding cost on the path towards node N_j (where node N_k is chosen as the next hop) by adding the current cost between itself and N_k and calculate the updated average cost using equation 6. It then re-evaluates the path to choose the optimal route to the destination. This process compares among all possible candidate routes and chooses the path that has the minimal average cost. If more than one candidate paths have same minimum average cost or have a difference of cost less than a given threshold η, the routing algorithm selects the path that has the least standard

Algorithm 1. Route Discovery in Pervasive Computing

```
Description: Route Discovery procedure simplifies the modified Distance Vector algorithm.
Input: destination $N_j$, reachable from node $N_k$
Output: : The routing table of a given source node S
Initialization:
Initialize cost to all nodes $N_j$ known to S to $\infty$
Calculate the trust between S and its immediate adjacent node $N_k$ in S's neighbor list
Add all immediate adjacent nodes to the routing table
for all node $N_k$ in the neighbor list do
    compute trust between S and $N_k$ (equation 5)
    compute average cost $D^S(N_k, N_k)$ (equation 6)
    compute running sum $DT(S, N_k)$
    compute running sum of squares $DT^2(S, N_k)$
end for
Iteration:
Wait until S detects change from its immediate link $N_k$ or receives a routing packet from its
neighbor /* This packet contains the information about the destination node $N_j$ */
if S detects change in its immediate link then
    Update the cost and propagate the change to all neighbors
else
    if $N_j$ is a destination that S has never seen before then
        Compute routing cost to $N_j$
        Compute running sum, running sum of squares, and hop count to $N_j$
        Add $N_j$ and its routing parameters into the routing table
    end if
    if $N_j$ is already in the routing table then
        Compute the routing cost to $N_j$
        Update the routing table if new cost is better than the current cost in the routing table
        Announce the new routing table to neighbors
    end if
end if
```

deviation as an optimal path. The standard deviation is calculated using the equation 7. Algorithm 1 gives the protocol used in generating the routing table. It consists of two phases: table initialization and iteration. The table initialization phase establishes paths to all immediate neighbors known to the source S. For each neighbor N_k, node S keeps track of hop count, average cost (calculated from equation 5), running sum of cost ($DT(S, N_k)$), and running sum of square of cost ($DT^2(S, N_k)$). These cost parameters are used for calculating the average cost and standard deviation according to the equation 6 and equation 7. The iteration phase is only triggered upon receiving the routing packets or upon changing of an immediate link with its neighbor.

In the first case, if the destination node N_j in the received packet is not known by node S, it will add N_j to the routing table and compute the routing cost to N_j by adding its trust between itself and its neighbor node who has sent the routing information of N_j to S. The routing cost to the destination N_j is computed as:

$$D^S(N_j, N_k) = \frac{v(S \longrightarrow N_k) + DT(N_k, N_j)}{hop\_cnt(N_k, N_j) + 1} \qquad (8)$$

where Node N_k is the sender of the routing information and $DT(N_k, N_j)$ is the forwarding cost from N_k to N_j. If S already knows the destination node N_j, it recomputes the routing cost to N_j and compares this value with the existing value. If the new cost is less or more stable[1] than the current cost, the cost to the destination N_j is updated. If the trust value between node S and its immediate neighbor N_k has changed, S has to recompute the routing cost to all destinations N_j where N_k is the next hop. The above equation 8 is used to recomputing the new routing cost. Then S compares the new cost to the current cost that S has in its routing table. If the new cost is less or more stable than the current cost, the cost to the destination N_j is updated.

5 Analysis

5.1 Security Analysis

The trust-based approach to routing is intended to minimize the effect of malicious nodes in the network. We discuss how the proposed scheme can reduce this effect. A malicious node can subvert the network in two ways:

Dropping packets. A malicious node on a path can deliberately drop the legitimate packets. Suppose a node N_i is sending a packet to N_j through the neighbor N_m who is malicious and drops packets arbitrarily. With every drop of packet, N_i increases the 'trust-negative packet forwarding counter' PFC_n corresponding to the node N_m in the interaction table. Note, in our scheme N_i cannot differentiate between a deliberate drop of packet and a packet drop due to valid reasons (like broken link or downtime of a node). However for a malicious node, number of dropped packets will be high compare to number of forwarded packets. This will lower the ratio $\frac{PFC_p - PFC_n}{PFC_p + PFC_n}$ and consequently N_i's trust on N_m will be low. Even if N_m keeps oscillating packet drop behavior, the above ratio will be close to zero and does not help N_m to increase its trust. Also note that N_m cannot fool N_i by dropping the actual data packet but forwarding the ECHO message to N_j. Because N_j will not reply to the ECHO message unless it receives the corresponding data packet.

Providing false recommendation. A malicious node N_m can disrupt the proper functioning of the scheme by providing false recommendation about a node. Suppose the malicious node N_m provides a "positive" rating about nodes N_k, N_l where both of them are malicious nodes. Every time N_i encounters a trust-negative packet forwarding event with any of them, N_i increases the 'trust-negative rating counter' RC_n of N_m which lowers the ratio $\frac{RC_p - RC_n}{RC_p + RC_n}$. Therefore, even if N_m behaves properly in the context of packet forwarding, it cannot subvert the system by falsely 'campaigning' for some other malicious nodes in the network. This also reduces the effect of collusion of malicious nodes

[1] This is used in the case when the new cost is equal to the the current cost or has a slight difference. The path that has less standard deviation is said to be more stable path.

to disrupt reliable routing. Similar action prevents the problem of 'badmouthing' i.e., when N_m provides false 'negative' rating about a 'good' node. This is possible because the trust of the benign node is not dependent just on rating provided by N_m, but involves other parameters on which N_m cannot have any control.

The above discussions show that the proposed trust-based routing scheme can reduce the effect of malicious nodes – working as individual or as a part of collusion, in attacks like arbitrary packet drops, false data injection and badmouthing.

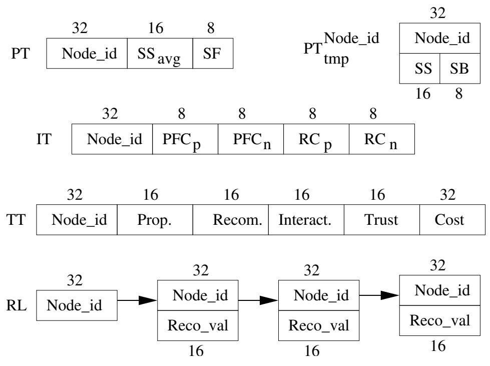
5.2 Complexity Analysis

As mentioned earlier, a typical device in a pervasive computing environment has relatively low resources in terms of storage and power. However, it needs to do some computations to evaluate the trust and cost. It also needs to store some values for a specific duration. In this section we analyze the computation, communication, and storage complexity of our protocol.

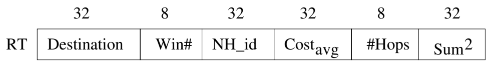
Computation complexity. In our approach the run-time complexity of the routing algorithm is not affected to a great extent. Our scheme only changes the metric of computing the administrative distance between nodes in the pervasive computing environment. It requires two additional computations for the running sum and the standard deviation. These additional computations do not change the big-O complexity of running time of the routing protocol as they are proportional to the size of the network. However, our approach requires additional methods in order to evaluate the trust between nodes. All these computations are simple arithmetic computations and have linear bounds. Consequently these do not add much to the computation overhead of the proposed protocol.

Communication complexity. A node maintains view of connectivity and trust relationship by periodically transmitting a beacon packet. This packet carries the node information including sequence number, hardware address, hardware protocol, and trust recommendation upon request. According to this scheme, there are 3 types of the beacon message – announcement, recommendation request, and recommendation reply. When a node receives a beacon message, it identifies the sender, measures the strength of the beacon signal (section 3.1), collects the trust information and recalculates the trust value of the sender. If the beacon message is a recommendation reply and the trustee is recognized by the receiver, it recomputes the trust value of the trustee with this recommendation information about the trustee. If the beacon message is a recommendation request message, and the trustee is recognized by the receiver, it prepares the trust value of the trustee and sends it to the requester with the next beacon message. We have estimated that a message 256 bits (32 bytes) long is sufficient to carry the beacon message. In the following discussion we show that routing tables in our protocol require only 154 bits for each record. Therefore, for a small network (say, with 50 nodes), the nodes pass a routing table which is of size less than 1KB.

Storage complexity. In our protocol each node has to maintain a certain number of tables. In this section we discuss the storage overhead that is required to store these tables. For each neighbor, a truster node needs to maintain the following tables and list: PT (properties table), $PT_{tmp}^{node\_id}$ (temporary property table), IT (interaction table), TT (trust table), and the list RL. In any table the node_id field takes 32 bits to store the address. We express the values of SS_{avg} in PT, SS in $PT_{tmp}^{node\_id}$, and *Properties, Recommendation, Interactions, Trust* in TT using 16 bits in which the most significant bit is the sign bit, the next bit expresses the exponent, and the rest 14 bits expresses the fraction. We need only 1 bit to express the exponent as all these numbers are within $[-1,1]$. Also, we get the precision of $1/2^{14}$ for trust related values. In the interaction table IT we use 8 bits to express each counter. The signaling factor is also expressed using 8 bits. However, we use 32 bits to represent the *cost* in TT as it is not bounded. Since the cost is always positive, we use first 16 bits for the exponent and the last 16 bits for the fraction. This gives the precision of $1/2^{16}$ for the cost value which is accurate enough for the environment. The figure 3(a) shows the structure of the tables stored for each neighbor. From the figure we see that the tables PT, IT, TT require $56, 64$ and 128 bits respectively for each record. Each of these tables is maintained for all the neighbors. If a node has m neighbors, then it requires $m \times 228$ bits. To store each signal and stability information we need 24 bits. If we assume that a node receives s signals within a window, then for each neighbor it needs $32 + s \times 24$ bits to store the signal. Hence for all m neighbors the node requires $m.(32 + s.24)$ bits. For the recommendation list RL each record requires 48 bits and we assume at most k $(k \leq m)$ recommenders recommend a neighbor. Therefore for each neighbor it requires $32 + k \times 48$ bits and hence for m neighbors $m.(32 + k.48)$ bits. Note, we have not considered the pointer size here as it depends on the implementation. Hence for each neighbor a node requires $228 + (32 + 24.s) + (32 + k.48) = 292 + 24.s + 48.k$ bits and for all m neighbors



(a) Tables maintained for each neighbor



(b) Routing table of the node

Fig. 3. Storage structure of the tables maintained in each node

total storage required is $m.(292+24.s+48.k)$ bits. This storage is not significant if we assume that each node in the pervasive computing environment interacts with a small number of neighbors. For example, in a pervasive computing environment topology where each node interacts with at most 20 neighbors and at most 100 signals are received within a time window *win*, then maximum number of bits required to store the trust information is $(292+24\times100+48\times20)\times20 = 3652\times20 = 73040$ bits $\approx$ 9KB.

Each node also stores a routing table whose structure is shown in the figure 3(b). The *Win#* field keeps the last window number. *#Hops* stores the number of hops to the destination. NH_id field stores the address of the next hop towards destination. Cost$_{avg}$ and $\sum^2$ fields store the cost metrics which are used in the route selection protocol. We also need the metric $\sum$ for route selection. However we do not store this information in the routing table as it can be derived from $Cost_{avg}$ and $\#Hops$. Each record of this table requires 154 bits and hence size of the routing table for each node in the topology is $O(154n)$ where n is the number of nodes in the network. In our example if we assume a small network with 50 nodes then each node require maximum 154×50 bits which is $\approx$ 0.9KB. Therefore all together a node requires only 10KB storage space to store the information related to trust-based routing.

The above analyses show that our protocol is light-weight in terms of trust evaluation, feedback management, message passing, and storage, thereby making it suitable for pervasive computing environment.

6 Conclusion and Future Work

In this work, we address the problem of reliable delivery of event data in pervasive computing environments to appropriate action points in order to support obligation policies. The problem is modeled as a routing problem. We present a trust-based approach to routing. Each node measures trustworthiness of its neighbor based on the neighbor's properties like signal strength and signal stability, a neighbor's behavior in forwarding packets from the node as well as in recommending other nodes, and a neighbor's rating by other neighbors. We represent each link in the network as a trust relationship with a numeric value between $[-1,1]$. This trust metric reflects reliability of a node as a packet forwarder. We have proposed a cost metric, which is inversely related to the trust metric, for each link in the network. We next adapt the distance vector routing algorithm for routing in pervasive computing environments. The modified algorithm uses the cost value assigned to each link to find the minimum average cost path from source to destination. We have discussed how our protocol can reduce the effect of malicious nodes. We have also shown that the scheme does not enhance the computation, communication, and storage overhead to any significant extent.

A lot of work still remains to be done. The proposed scheme is generic in nature. We need to modify specific ad hoc routing protocols using our metrics and compare the results to evaluate the performance of the proposed scheme. We plan to run simulation experiments to compare the routing results with existing ad hoc routing protocols. We are also looking for other node properties like remaining battery power to extend the 'properties' parameter of the trust metric.

References

1. Balfanz, D., Smetters, D., Stewart, P., Wong, H.: Talking to Strangers: Authentication in Adhoc Wireless Networks. In: Symposium on Network and Distributed Systems Security (NDSS '02), San Diego, California, USA (February 2002)
2. Chiang, C.C., Wu, H.K., Liu, W., Gerla, M.: Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel. In: 5th IEEE Singapore International Conference on Networks (SICON'97), Kent Ridge, Singapore pp. 197–211 (April 1997)
3. Corson, M.S., Ephremides, A.: A Distributed Routing Algorithm for Mobile Wireless Networks. Wireless Networks 1(1), 61–82 (1995)
4. Gafni, E., Bertsekas, D.: Distributed Algorithms for Geneerating Loop-free Routes in Network with Frequently Changing Topology. IEEE Transaction and Communication 29(1), 11–15 (1981)
5. Gerla, M., Tsai, J.T.: Multicluster, Mobile, Multimedia Radio Network. Wireless Networks 1(3), 255–265 (1995)
6. Hu, Y.C., Perrig, A., Johnson, D.B.: Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In: Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom'02), Atlanta, Georgia, USA (September 2002)
7. Papadimitratos, P., Haas, Z.: Secure Data Transmission in Mobile Ad Hoc Networks. In: ACM Workshop on Wireless Security (WiSe'03), San Diego, California, USA (September 2003)
8. Perkins, C.E., Bhagwat, P.: Highly Dynamic Destination-Sequenced Distance Vector (DSDV) for Mobile Computers. In: Conference on Communication Architectures, Protocols and Applications (SIGCOMM'94), London, UK, pp. 234–244 (August 1994)
9. Toh, C.K.: A Novel Distributed Routing Protocol To Support Ad hoc Mobile Computing. In: IEEE 15th Annual International Phoenix Conference on Computers and Communication (IPCCC'96), Phoenix, AZ, USA, pp. 480–486 (1996)
10. Zhou, L., Haas, Z.J.: Securing Ad Hoc Networks. IEEE Network 13(6), 24–30 (1999)
11. Zouridaki, C., Mark, B.L., Hejmo, M., Thomas, R.K.: A Quantitative Trust Establishment Framework for Reliable Data Packet Delivery in MANETs. In: Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'05), Alexandria, VA, USA, pp. 1–10. ACM Press, NewYork (2005)
12. Awerbuch, B., Holmer, D., Nita-Rotaru, C., Rubens, H.: An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In: ACM Workshop on Wireless Security (WiSe'02), Atlanta, GA, USA, pp. 21–30 (September 2002)
13. Dube, R., Rais, C.D., Wang, K.Y., Tripathi, S.K.: Signal Stability-Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks. IEEE Personal Communications Magazine 4(1), 36–45 (1997)
14. Yi, S., Naldurg, P., Kravets, R.: Security-Aware Ad Hoc Routing for Wireless Networks. In: Proceedings of the 2nd ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001), Long Beach, CA, October 2001, pp. 299–302. ACM Press, New York (2001)
15. Ray, I., Chakraborty, S.: A Vector Model of Trust for Developing Trustworthy Systems. In: Samarati, P., Ryan, P.Y. A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 260–275. Springer, Heidelberg (2004)

Privacy-Preserving Schema Matching Using Mutual Information*

Isabel F. Cruz[1], Roberto Tamassia[2], and Danfeng Yao[2]

[1] Department of Computer Science
University of Illinois at Chicago
ifc@cs.uic.edu

[2] Department of Computer Science
Brown University
{rt,dyao}@cs.brown.edu

The problem of *schema or ontology matching* is to define *mappings* among schema or ontology elements. Such mappings are typically defined between two schemas or two ontologies at a time. Ideally, using the defined mappings, one would be able to issue a single query that will be rewritten automatically to all the databases, instead of manually writing a query to each database. In a centrally mediated architecture a query is written in terms of a global schema or ontology that integrates all the database schemas or ontologies, while in a peer-to-peer architecture a query is written in terms of the schema or of the ontology of any of the peer databases.

Automatic schema matching approaches can use only the schema, only the instances, or a combination of both. Mappings can take into account not only concept properties (e.g., string similarity), but also constraints (e.g., relationship cardinality) and schema structure (e.g., graph similarity) [9].

Security and privacy issues arise in the context of data integration. For example, previous work looks into secure access to mediated data [2,4]. Other work has defined the concept of *minimal necessary information sharing* that applies to querying: in computing the answer to a query, only the query result should be revealed [1]. Most matching approaches rely on the fact that both schemas or ontologies are completely visible by both parties. Clearly, this approach disregards security and privacy considerations. Even within the same organization, different users have access to different database views. It is, therefore, only natural to create automatic mechanisms by which mappings can be established between a pair of schemas or ontologies, without each party needing to reveal their whole metadata.

Clifton *et al.* discuss issues and identify research directions in privacy-preserving data integration, including those that arise in schema matching [3]. More recently, Mitra *et al.* look at the specific issue of privacy-preserving ontology matching [7,8]. In their approach, terms in the ontologies and in the matching rules (which define the mappings) are encrypted, so that the mediator does not see the actual terms. However, during the ontology matching process, which is semi-automatic, a human expert has access to both ontologies in cleartext (using a session key).

We propose an automatic privacy-preserving schema matching protocol. The result of this protocol is the set of mappings between attributes in the schemas of the two

* This work was supported in part by the National Science Foundation under ITR awards IIS–0326284, IIS–0324846, and IIS–0513553.

intervening parties. Most importantly, from a privacy-preserving viewpoint, we do not use a third-party mediator and only those schema attributes that are matched are revealed by a party to the other party.

Our approach to privacy-preserving schema matching is based on the instance-based schema matching approach by Kang and Naughton [6], which considers the dependencies among data instances, as measured by the mutual information among every pair of attributes in each schema. For each schema, these dependencies are represented as a weighted graph and matching between the two schemas relies on matching the corresponding graphs. The mutual information between two attributes is a measure of the amount of information that each attribute contains about the other attribute. Mutual information can be computed using the entropies of the individual attributes and the conditional entropies. We consider three types of mappings: one-to-one, onto, and partial.

We develop an efficient privacy-preserving schema matching protocol using mutual information of pair-wise attributes. The protocol is executed by two entities, each having a private schema. The output of the protocol is a set of mappings between the matching attributes of the two schemas. We prove that our privacy-preserving schema matching protocol is secure against malicious adversaries for all mapping types. One of the building blocks of our protocol is the privacy-preserving set intersection scheme by Freedman, Nissim, and Pinkas [5]. We show that in the case where all the attribute entropies in one of the schemas are different from one another, the protocol executes a linear number of privacy-preserving set intersections.

References

1. Agrawal, R., Evfimievski, A.V., Srikant, R.: Information sharing across private databases. In: Proc. ACM SIGMOD, pp. 86–97. ACM Press, New York (2003)
2. Candan, K.S., Jajodia, S., Subrahmanian, V.S.: Secure mediated databases. In: Proc. IEEE Int. Conf. on Data Engineering, pp. 28–37. IEEE Computer Society Press, Los Alamitos (1996)
3. Clifton, C., Kantarcioglu, M., Doan, A., Schadow, G., Vaidya, J., Elmagarmid, A.K., Suciu, D.: Privacy-preserving data integration and sharing. In: Proc. ACM Workshop on Research Issues in Data Mining and Knowledge Discovery, pp. 19–26. ACM Press, New York (2004)
4. Dawson, S., Qian, S., Samarati, P.: Providing security and interoperation of heterogeneous systems. Journal of Distributed and Parallel Databases 8(1), 119–145 (2000)
5. Freedman, M., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
6. Kang, J., Naughton, J.F.: On schema matching with opaque column names and data values. In: Proc. ACM SIGMOD, pp. 205–216. ACM Press, New York (2003)
7. Mitra, P., Liu, P., Pan, C.-C.: Privacy-preserving ontology mapping. In: Proc. Int. Workshop on Contexts and Ontologies: Theory, Practice and Applications (2005)
8. Mitra, P., Pan, C.-C., Liu, P., Atluri, V.: Privacy-preserving semantic interoperation and access control of heterogeneous databases. In: Proc. ACM Conf. on Computer and Communications Security, pp. 66–77. ACM Press, New York (2006)
9. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal 10(4), 334–350 (2001)

The Interval Revocation Scheme for Broadcasting Messages to Stateless Receivers

Anna Zych, Milan Petković, and Willem Jonker

Philips Research, Eindhoven, The Netherlands
anusiek@gmail.com,{milan.petkovic,willem.jonker}@philips.com

The Broadcast Encryption methods, often referred to as revocation schemes, allow data to be efficiently broadcast to a dynamically changing group of users. A special case is when the receivers are stateless [2,1]. Naor et al. [2] propose the Complete Subset Method (CSM) and the Subset Difference Method (SDM). Asano [1] puts forth two other methods, AM1 and AM2, which use public prime parameters to generate the decryption keys. The efficiency of broadcast encryption methods is measured by three parameters: (i) message size - the number of transmitted ciphertexts; (ii) storage at receiver - the number of private keys each receiver is required to store; and (iii) key derivation time - the computational overhead needed to access the decryption keys.

Let $\mathcal{N} = \{u_0, ..., u_{N-1}\}$ be the set of N receivers and $\mathcal{R} \subset \mathcal{N}$ be a group of r users whose decryption privileges should be revoked. The aim of a revocation scheme is to allow a transmission of a message M to all users in such a way, that any user $u \in \mathcal{N} \setminus \mathcal{R}$ can decrypt the message correctly, while even a coalition consisting of all members of $\mathcal{R}$ can not decrypt it.

We propose a new revocation scheme for transmitting secret messages to stateless receivers. In comparison to other schemes, our scheme improves private storage to one key per receiver and the size of the message to the number of revoked receivers r, while the time needed for deriving a key is of order of a logarithm of the number of all receivers $O(\log N)$. We push the storage requirements to the public space of N^2 parameters that are needed to derive the keys. We provide the comparison of CSM, SDM, AM1 and AM2 methods with our method in Table 1.

Table 1. Performance of methods in [2,1]

	CSM [2]	SDM [2]	AM1 [1]	AM2 [1]	Our method
Message size	$r \log \frac{N}{r}$	$2r - 1$	$r(\frac{\log \frac{N}{r}}{loga} + 1)$	$r(\frac{\log \frac{N}{r}}{loga} + 1)$	r
Storage at rec.	$\log N$	$\frac{\log^2 N}{2}$	1	$\frac{\log N}{\log a}$	1
Key der. time	-	$O(\log N)$	$O(\frac{(2^{a-1}-1)\log N}{\log a})$	$O(2^{a-1} - 1)$	$O(\log N)$

A typical revocation scheme (compliant to the framework provided in [2]) defines a collection of subsets $\mathcal{X} = S_1, ..., S_w$, $S_j \subseteq \mathcal{N}$. Each subset S_j is assigned a long-lived secret key K_j. Each user $u \in S_j$ should be able to deduce K_j from secret information assigned to her during the initiation phase. Deducing K_j however should be infeasible for any coalition of users $\{u_1 \dots u_t\} \subset \mathcal{N} \setminus S_j$. Given a revoked set $\mathcal{R}$, the remaining

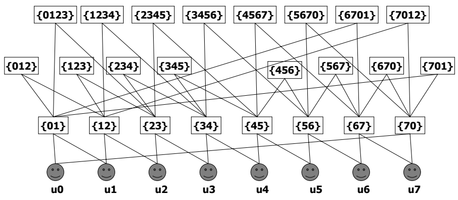


Fig. 1. Example of a digraph restricted to intervals of size $1 \ldots 4$

users $\mathcal{N} \setminus \mathcal{R}$ are partitioned into $S_{i_1}, \ldots, S_{i_m}$ so that $\mathcal{N} \setminus \mathcal{R} = \bigcup_{j=1}^{m} S_{i_j}$ and a session key K is encrypted m times with (hash values) of $K_{i_1}, \ldots K_{i_m}$. Such header is broadcasted together with the content encrypted with the session key. In the scheme's initiation phase, every receiver u is assigned private information $I[u]$, which allows to compute K_j for each group S_j such that $u \in S_j$.

Thus, a particular scheme is specified by the collection of subsets $\mathcal{X}$, a method to assign the keys to each subset of the collection, a method to cover non-revoked receivers $\mathcal{N} \setminus \mathcal{R}$ and a method that allows each user $u \in S_j$ to compute her key K_j from $I[u]$.

We propose here the interval revocation scheme. An interval $I \subset \mathcal{N}$ is a subset of $\mathcal{N}$ containing consecutive elements: $I[i,j] = \{u_{(i \mod N)}, u_{(i+1 \mod N)}, ..., u_{(j \mod N)}\}$ For example for $N = 6$ interval $I[2,5] = \{u_2, u_3, u_4, u_5\}$, but interval $I[2,1] = \{u_2, u_3, u_4, u_5, u_0, u_1\}$. The size of an interval is $|I[i,j]| = j - i + 1 \mod N$. Interval $I[i, i+s-1]$ of size s can be split uniquely into two intervals of size $\ulcorner \frac{s}{2} \urcorner$ as follows: $I[i, i+s-1] = I[i, i + \ulcorner \frac{s}{2} \urcorner - 1] \cup I[i + \ulcorner \frac{s-1}{2} \urcorner, i+s-1]$ (1).

We define collection $\mathcal{X}$ as the collection of all intervals on $\mathcal{N}$. Based on (1), each interval $I \in \mathcal{X}$ of size s can be uniquely split into $I_{left}, I_{right} \in \mathcal{X}$ of size $\ulcorner \frac{s}{2} \urcorner$. Furthermore, any two intervals never share the same set of children. A digraph representing the child relation for $N = 8$ is presented in Figure 1, restricted to intervals of size $1, 2, 3$ and 4. Let $\mathcal{R} = \{u_{i_1}, u_{i_2}, ..., u_{i_r}\} \subset \mathcal{N}$ be the set of revoked receivers. The cover of $\mathcal{N} \setminus \mathcal{R}$ consists of all intervals between revoked receivers. We have: $\mathcal{N} \setminus \mathcal{R} \subset I[i_r + 1, i_1 - 1] \cup \bigcup_{j=1, i_{j+1} > i_j + 1}^{r-1} I[i_j + 1, i_{j+1} - 1]$, and we define the cover as the set of intervals from this sum. Thus, the size of the cover is at most $r = |\mathcal{R}|$.

We apply the Diffie - Hellman (DH) key exchange protocol for key derivation. We label each interval $I \in \mathcal{X}$ with its private key S_I and its public key P_I. The key of interval I is a shared key obtained by applying the DH protocol on the private and public keys of its children I_{left} and I_{right}, treating children as the key exchanging parties. To derive a key of a descendant interval, a receiver needs his own secret key, as well as the public keys of the "other" children in the path to the target interval. Receiver u_i needs to store only the secret key S_I assigned to interval $I = I[i,i]$. The number of operations needed to derive one key from another is $O(\log N)$.

Given the achieved results, the direction for future research is to find an assignment of public parameters that can be generated efficiently in on-the-fly manner. This would allow to release the public space requirement for our scheme.

References

1. Asano, T.: A revocation scheme with minimal storage at receivers. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 433–450. Springer, Heidelberg (2002)
2. Naor, D., Naor, M., Lotspiech, J.B.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)

Measuring the Overall Security of Network Configurations Using Attack Graphs

Lingyu Wang[1], Anoop Singhal[2], and Sushil Jajodia[3]

[1] Concordia Institute for Information Systems Engineering
Concordia University
Montreal, QC H3G 1M8, Canada
wang@ciise.concordia.ca
[2] Computer Security Division, NIST
Gaithersburg, MD 20899, USA
anoop.singhal@nist.gov
[3] Center for Secure Information Systems
George Mason University
Fairfax, VA 22030-4444, USA
jajodia@gmu.edu

Abstract. Today's computer systems face sophisticated intrusions during which multiple vulnerabilities can be combined for reaching an attack goal. The overall security of a network system cannot simply be determined based on the number of vulnerabilities. To quantitatively assess the security of networked systems, one must first understand which and how vulnerabilities can be combined for an attack. Such an understanding becomes possible with recent advances in modeling the composition of vulnerabilities as *attack graphs*. Based on our experiences with attack graph analysis, we explore different concepts and issues on a metric to quantify potential attacks. To accomplish this, we present an *attack resistance* metric for assessing and comparing the security of different network configurations. This paper describes the metric at an abstract level as two composition operators with features for expressing additional constraints. We consider two concrete cases. The first case assumes the domain of attack resistance to be real number and the second case represents resistances as a set of initial security conditions. We show that the proposed metric satisfies desired properties and that it adheres to common sense. At the same time, it generalizes a previously proposed metric that is also based on attack graphs. It is our belief that the proposed metric will lead to novel quantitative approaches to vulnerability analysis, network hardening, and attack responses.

1 Introduction

Today's networked computer systems constitute the core component of information technology infrastructures in enterprises and in critical infrastructures, such as power grids, financial data systems, and emergency communication systems. Protecting such systems against malicious intrusions is crucial to the economy and to our national security. Having a standard way for measuring various aspects of network security will bring together users, vendors, and labs in specifying, implementing, and evaluating the requirements and features of network security products. However, in spite of various efforts in standardizing security metric, a widely-accepted metric for network security is still largely unavailable. This is partly due to the fact that most researchers are still adopting a qualitative and imprecise view toward the evaluation of network security. For example, typical issues addressed in current research may ask following questions. Are all critical resources in a network secure (topological vulnerability analysis)? Can a network be hardened to secure the given resources (network hardening)? How to stop an ongoing intrusion from compromising given resources (attack response)?

The qualitative nature of these questions reflect the current focus on the qualitative, rather than quantitative, study of network security. This focus implies the inherent impreciseness in many research results and also indicates the need for more research efforts on security metrics. However, the lack of research on quantitative aspects of network security is natural. Assessing the overall security of a network requires a thorough understanding of the interplay between host vulnerabilities. That is, which and how vulnerabilities can be combined for an attack. Such an understanding is difficult to obtain with existing security tools, such as vulnerability scanners and intrusion detection systems. These tools typically focus on identifying individual vulnerabilities or attacks, and are usually unaware of the relationships among vulnerabilities or attacks.

Recent advances in modeling compositions of vulnerabilities using *attack graphs* (a review of related work will be given in the next section) indicate that the research has progressed to a point where the quantitative study of network security is critical and, at the same time, possible. Attack graphs supplement vulnerability scanners with the missing information about relationships among vulnerabilities. Analyzing the correlated vulnerabilities thus provides a clear picture about what attacks might happen in a network and about their consequences. Attack graphs thus allow us to consider potential attacks in a particular *context* relevant to the given network. The current work is based on our past experiences with attack graph analysis [12,15,16,21,29,30,31,32] and a practical tool, the Topological Vulnerability Analysis (TVA) system, with the capability of modeling more than 37,000 vulnerabilities taken from 24 information sources including X-Force, Bugtraq, CVE, CERT, Nessus, and Snort [12]. The presence of such a powerful tool demonstrates the practicality of using attack graphs as the basis for measuring network security.

Instead of measuring individual vulnerabilities and then wondering about their combined effect, this paper measures the overall security of a network using the context provided by an attack graph. Such a capability will enable us to answer important questions like (but not limited to): How much effort and time will it take to compromise a critical resource under each possible network configuration? Answers to such questions will allow system administrators to choose the optimal configuration that is the most resistant to potential attacks. More specifically, we propose an *attack resistance* metric for assessing and comparing the security of different network configurations. The metric is based on intuitive properties derived from common sense. For example, our metric will indicate reduced security when more attack paths exist, whereas it indicates increased security for longer and more difficult paths. To make the metric broadly applicable, we first describe it at an abstract level as two composition operators with functions that allow for expressing additional dependency relationship between resistances. We then consider two concrete cases. The first assumes the domain of attack resistance to be real number and the second represents resistances as sets of initial security conditions. For the first case, we propose to use operators that are analogous to the ones used in computing the resistance of a series and parallel circuit. We study additional issues that arise due to the unique properties of attack graphs. For the second case, we show that a previously proposed metric [21] is equivalent to our metric under certain conditions.

The rest of the paper is organized as follows. Section 2 outlines a framework for defining security metrics using attack graphs. Section 3 presents the attack resistance metric. Section 4 reviews related work. Finally, Section 5 concludes the paper.

2 A Framework for Defining Security Metrics Using Attack Graphs

This section first reviews the attack graph model and then discusses intuitions behind the proposed metric.

2.1 Attack Graph Model

We adopt the attack graph model used in the Topological Vulnerability Analysis tool [12], which is one of the most advanced utilities for generating and analyzing attack graphs. This attack graph model is similar in nature to the earlier ones based on modified model checking [26], but it avoids the potential combinatorial explosion faced by the latter. More specifically, it makes a *monotonicity assumption* stating an attacker never relinquishes an obtained capability [1]. An attack graph can thus record the dependency relationship between exploits instead of recording all attack paths. The resulting attack graph has no duplicate vertices and hence has a polynomial size in the number of vulnerabilities multiplied by the number of connected pairs of hosts.

In our model, an *Attack graph* is a directed graph representing prior knowledge about vulnerabilities, their dependencies, and network connectivity. The vertices of an attack graph are divided into two categories, namely, *exploits* and *security conditions* (or simply *conditions* when no confusion is possible). First, exploits are actions taken by attackers on one or more hosts in order to take advantage of existing vulnerabilities. We denote an exploit as a predicate. For example, an exploit involving three hosts can be denoted using $v(h_s, h_m, h_d)$, which indicates an exploitation of the vulnerability v on the destination host h_d, initiated from the source host h_s, through an intermediate host h_m. Similarly, we write $v(h_s, h_d)$ or $v(h)$, respectively, for exploits involving two hosts (no intermediate host) or one (local) host.

Second, a security condition is a property of the system or network that is relevant to some exploits. A condition is relevant to an exploit if it is either required for executing the exploit or satisfied by executing the exploit. We also use a predicate to represent a condition involving one or more hosts. For example, $c(h_s, h_d)$ indicates a security-related condition c involving the source host h_s and the destination host h_d. Similarly, a condition that only involves a single host can be written as $c(h)$. Examples of security conditions include the existence of a vulnerability, the existence of network connectivity or trust relationship between two hosts. It is worth noting that an attack graph usually includes exploits and conditions corresponding to normal services or functionality. Such services are included because they may help attackers in escalating their privileges when combined with other exploits, although they are not intended for that purpose. On the other hand, this fact also implies that not all exploits can be removed in hardening a network, so measuring the relative security of different configurations becomes important.

Directed edges in an attack graph inter-connect exploits with conditions. No edge directly goes between two exploits or between two conditions. First, an edge from a condition to an exploit denotes the *require* relation, which means the exploit cannot be executed unless the condition is satisfied. Second, an edge pointing from an exploit to a condition denotes the *imply* relation, which means executing the exploit will satisfy the condition. For example, an exploit typically requires at least two conditions, that is the existence of the vulnerability (which could be a normal service) on the destination host and the network connectivity between the two hosts. We formally characterize attack graphs in Definition 1.

Definition 1. *Given a set of exploits $\mathcal{E}$, a set of conditions $\mathcal{C}$, and two relations $require \subseteq \mathcal{C} \times \mathcal{E}$ and $imply \subseteq \mathcal{E} \times \mathcal{C}$, an* **attack graph** *$G$ is the directed graph $G(\mathcal{E} \cup \mathcal{C}, require \cup imply)$ ($\mathcal{E} \cup \mathcal{C}$ is the vertex set and $require \cup imply$ the edge set).*

One important semantics of attack graphs is that the require relation is conjunctive, whereas the imply relation is disjunctive. More precisely, an exploit cannot be realized until *all* of its required conditions have been satisfied, whereas a condition is satisfied if *any* of the realized exploits implies that condition. Sometimes only exploits in an attack graph are of interest, we thus remove conditions to obtain an *exploit dependency graph*. However, in such a graph, edges between exploits may represent both the conjunctive and disjunctive relationship. For example, in an attack graph, if two exploits e_1 and e_2 both imply the same condition c_1, which is required by another exploit e_3, then e_3 can be executed after executing either e_1 or e_2 (since c_1 will be satisfied by any of them). On the other hand, if e_1 implies c_1, e_2 implies a different condition c_2, and both c_1 and c_2 are required by e_3, then e_3 cannot be executed before both e_1 and e_2 are. We shall need this observation later in the paper.

2.2 Motivating Example

To build intuitions about properties that a security metric should satisfy, we consider the well-known attack scenario as shown on the left hand side of Figure 1 (notice that this is an overly simplified example for illustration purposes, and our metric and techniques are intended for more complicated cases where results cannot be obtained through observations). In this attack graph, exploits are depicted in ovals and conditions in clear text. The critical condition that needs to be guarded is shown in a shaded oval. The attack graph basically indicates that an attacker on host 0 can obtain user privilege on host 1, either using an SSH buffer overflow attack or through the trust relationship established by uploading the .rhost file through FTP. The attacker can use the latter trick to obtain user privilege on host 2, either directly from host 0 or using host 1 as an intermediate stepping stone. The attacking goal, that is the root privilege on host 2, can then be obtained using a local buffer overflow attack.

The right hand side of Figure 1 shows the exploit dependency graph. It is worth noting that in this specific case only disjunctive dependency relationship exists between

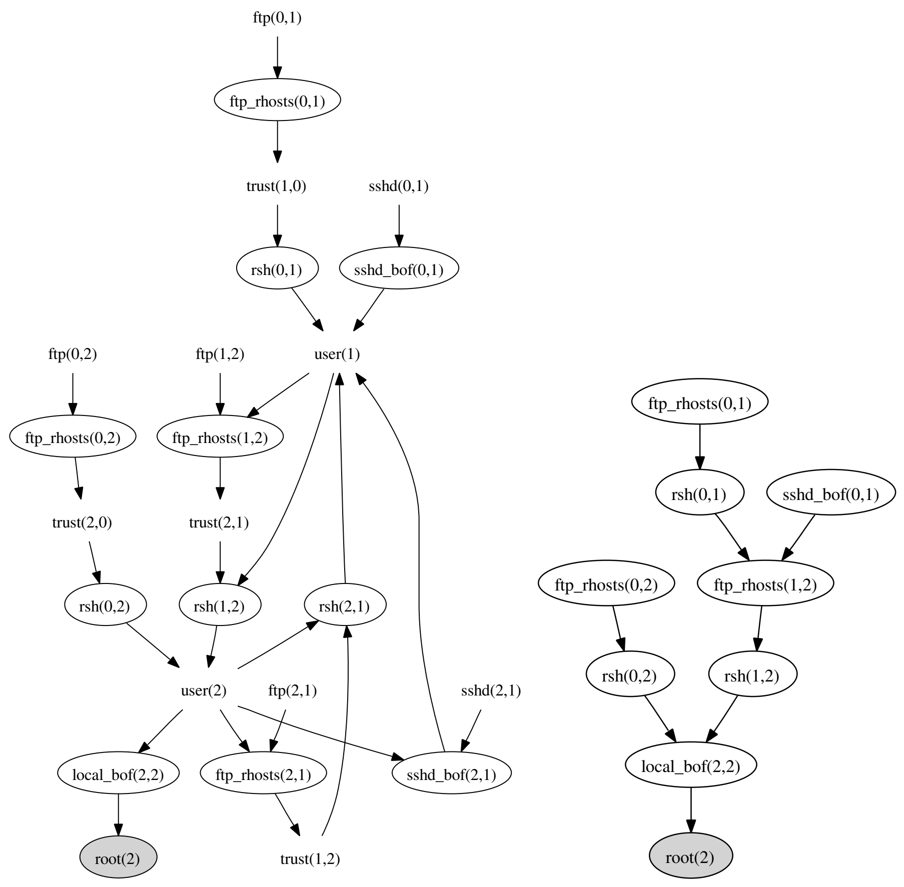


Fig. 1. An Example of Attack Graph and Exploit Dependency Graph

exploits. For example, there are two alternative ways to reach the exploit ftp_rhosts $(1,2)$ and similarly two ways to reach the exploit $local\_bof(2,2)$. In general, the dependency relationship between exploits can be both disjunctive and conjunctive, and the graph notation is thus not sufficient to distinguish between the two. Later we shall introduce a special notation for this purpose.

We make several observations in Figure 1. First, the two loops via exploit ftp_rhosts $(2,1)$ and $sshd\_bof(2,1)$ are both removed. These loops both allow the attacker to obtain user privilege on host 1 for the second time, after the privilege has already been obtained (otherwise the two exploits cannot be executed). An attacker can certainly make such redundant attacking effort at will, but a security metric should assume the most efficient attackers and indicate the security of a network in the worst-case scenario. That is, *a metric should never yield a value that is greater than the smallest attacking effort required for reaching the attack goal.*

Second, the exploits in Figure 1 clearly have different difficulty in terms of the time and effort required for their execution. For example, the ftp_rhosts and rsh exploits both take advantage of normal services in a clever way, and they usually do not require much time or effort if the attacker has the basic knowledge about the attack (between the two type of exploits, rsh may be slightly easier than ftp_rhosts in the sense that the latter requires crafting the .rhost file). On the other hand, both the $sshd\_bof$ and the $local\_bof$ are buffer overflow attacks, which require significantly more knowledge and time than the previous two because a buffer overflow attack usually requires brute force effort to determine proper parameters. This example thus shows *different exploits have different difficulties in terms of effort and time required for their execution*.

Third, there are three possible attack paths (that is, sequences of attacks) reaching the attack goal, as shown on the right hand side of Figure 1, the left path (that is, the one through $ftp\_rhosts(0,2)$ and $rsh(0,2)$) requires the smallest amount of effort. The middle path requires slightly more effort since it involves both host 1 and host 2. The right path demands the most effort because it requires an additional buffer overflow attack $sshd\_bof(0,1)$. Recall the above argument that security should be measured as the smallest effort required to reach the goal. It seems that the left path is a good candidate to be used as the measure of overall security. However, it is important to notice that when multiple paths coexist in an attack graph, reaching the attack goal is actually easier than if only one of these paths exists (even if the path requires the smallest amount of effort). Intuitively, more attack opportunities mean less security, because attackers will have a better chance to reach the attack goal. In this specific case, even though the middle and the right paths are more difficult than the left one, they nevertheless represent possibilities for attacks and thus they do reduce the overall security of the network. That is, *multiple attack paths together are less secure than any of the paths alone*.

Finally, assuming the middle attack path is followed by an attacker, it can be argued that the exploit $ftp\_rhosts(1,2)$ may be slightly easier than its predecessor ftp_rhosts $(0,1)$. To launch the same type of attack for the second time, the attacker will benefit from his/her experiences and tools that have been accumulated while launching the attack for the first time. It is, however, not possible to add an edge between these two exploits in attack graph, because the exploit $ftp\_rhosts(1,2)$ does not directly depend on $ftp\_rhosts(0,1)$ (with $rsh(0,1)$ in the middle). This implies that an additional relation is needed to encode such dependency relationship between exploits, which is different from the $imply$ or $require$ relations already encoded in attack graphs. In another word, *executing an exploit may change the difficulty of executing another exploit, even if the two do not directly depend on each other in the attack graph*.

The above requirements are largely common sense that should be satisfied by a security metric. The rest of the paper proposes a security metric based on the attack graph model by taking these requirements into consideration.

3 An Attack Resistance Metric

This section proposes an attack resistance metric based on the attack graph model. We first discuss the metric in a generic form. We then discuss two concrete cases to illustrate the metric in more details. We address various issues encountered while computing the metric from a given attack graph.

3.1 A Generic Framework

We propose to measure the *attack resistance* of a network configuration as the composition of measures of individual exploits. Ideally, the resistance of each type of exploits in terms of effort and time should be represented as a total order, such as using real numbers (the next section considers how individual resistances can be combined when attack resistance is represented as a real number). Unfortunately, although clearly desired, the information and resources required by this ideal situation are limited [17]. It is, however, usually possible to estimate an approximate ordering or a partial ordering on the domain of attack resistance. We shall also consider another case where the resistance of individual exploit is simply the set of initial conditions (that is, conditions not implied by other exploits).

Different applications may define the attack resistance of individual exploits in significantly different ways. To make our metric broadly applicable, we describe the metric in a generic form while leaving the individual measures uninterpreted. Central to the model are two types of composition operators, denoted as $\oplus$ and $\otimes$. The two operators correspond to the disjunctive and conjunctive dependency relationship between exploits in an attack graph, respectively. Based on the intuitive properties mentioned in Section 2.2, the two operators should satisfy that $r_1 \oplus r_2$ is no greater than r_1 or r_2, whereas $r_1 \otimes r_2$ is no less than r_1 and r_2, with respect to a given ordering on the domain of attack resistance.

In addition to the two composition operators, we introduce a function $\mathcal{R}()$ that maps a set of exploits to another exploit and its resistance value. The function is intended to capture a special kind of dependency relationship between exploits. That is, executing some exploits may affect the resistance value of another exploit, even though the latter cannot be executed yet. In most cases, this effect will be to assign a lower resistance value to the affected exploit. For example, exploits involving the same vulnerability should be related together using this function such that successfully exploiting one instance of the vulnerability reduces the resistance of others due to the attacker's accumulated experiences and tools. We shall also show that this function is useful in handling the non-tree structure of attack graphs. We summarize the model in Definition 2.

Definition 2. *Given an attack graph $G(\mathcal{E} \cup \mathcal{C}, require \cup imply)$ with attack goals $g \subseteq \mathcal{C}$, the attack resistance metric is composed of*

– *A total function $r() : \mathcal{E} \to \mathcal{D}$,*
– *a total function $R() : \mathcal{E} \to \mathcal{D}$,*
– *an operator $\oplus : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$,*
– *an operator $\otimes : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$, and*
– *a function $\mathcal{R}() : \mathcal{E} \to \mathcal{E} \times \mathcal{D}$.*

We call the set $\mathcal{D}$ the **domain** *of resistance, $r(e)$ the* **individual resistance** *(or simply resistance) of an exploit e, $R(e)$ the* **cumulative resistance** *of e.*

The main tasks in implementing this metric for a specific application is to populate the individual resistance by defining the function $r()$, to determine suitable operators $\oplus$ and $\otimes$, to capture additional dependency relationships between exploits using the function $\mathcal{R}$, and finally to decide how the cumulative resistance function $R()$ should be computed based on these information. The cumulative resistance of each attack goal then provides a quantitative measure as how likely that attack goal can be achieved, or equivalently, how vulnerable the corresponding resource is under a given network configuration.

3.2 Attack Resistance as Real Numbers

We now consider a concrete case where the domain of resistance $\mathcal{D}$ is the non-negative real number. Analogous to the resistance of a series and parallel circuit, we define $\oplus$ as the reciprocal of the sum of the reciprocal of individual resistance values. That is, $\frac{1}{r_1 \oplus r_2} = \frac{1}{r_1} + \frac{1}{r_2}$. The operator $\otimes$ is simply addition. Recall our discussions about the relative difficulty of different type of exploits in Section 2.2. Suppose we assign the value 10 to be the resistance of each $sshd\_bof$ and $local\_bof$, the value 2 and 1 to each ftp_rhosts and rsh exploit, respectively, as depicted on the left hand side of Figure 2. The cumulative resistances can then be computed as follows, where $r()$ stands for the individual resistance and $R()$ the cumulative resistance (we shall not consider the function $\mathcal{R}$ for the time being). The final results are shown in the right hand side of Figure 2.

- $R(rsh(0,1)) = r(ftp\_rhosts(0,1)) + r(rsh(0,1)) = 2 + 1 = 3$
- $R(ftp\_rhosts(1,2)) = 1/(1/R(rsh(0,1)) + 1/r(sshd\_bof(0,1)))+$ $r(ftp\_rhosts(1,2)) = 1/(1/3 + 1/10) + 2 \approx 4.3$
- $R(rsh(1,2)) = R(ftp\_rhosts(1,2)) + r(rsh(1,2)) \approx 4.3 + 1 = 5.3$
- $R(rsh(0,2)) = r(ftp\_rhosts(0,2)) + r(rsh(0,2)) = 2 + 1 = 3$
- $R(local\_bof(2,2)) = 1/(1/R(rsh(0,2)) + 1/R(rsh(1,2)))+$ $r(local\_bof(2,2)) = 1/(1/3 + 1/5.3) + 10 \approx 11.9$

According to our discussions in Section 2.2, the cumulative resistance of the whole network should be smaller than the cumulative resistance of each possible attack path. The cumulative resistance for each attack path reaching the goal can be computed by simply adding (that is, the $\otimes$ operator) individual resistance values along the path. The results for the three attack paths in Figure 2 are 13, 16, and 23, from left to right. Clearly, the accumulative resistance of the whole network, 11.9, is indeed smaller than any of the three values, satisfying the intuitive requirements given in Section 2.2. We may also notice that the composition (using the operator $\oplus$) of these three resistance values is about 5.5, which is less than the computed resistance 11.9. This reflects the fact that the three paths are not disjoint. The value 5.4 is computed under the implicit assumption that the three paths are disjoint, which is not the case here. Intuitively, having common exploits among different paths may increase the overall attack resistance, because the attacker must execute these exploits no matter what path they follow. Our metric naturally takes into consideration the overlapping portion of the paths. Above discussions also indicate that cumulative resistances can be computed in a breadth-first manner, which takes time $O(\mid \mathcal{E} \mid^2)$.

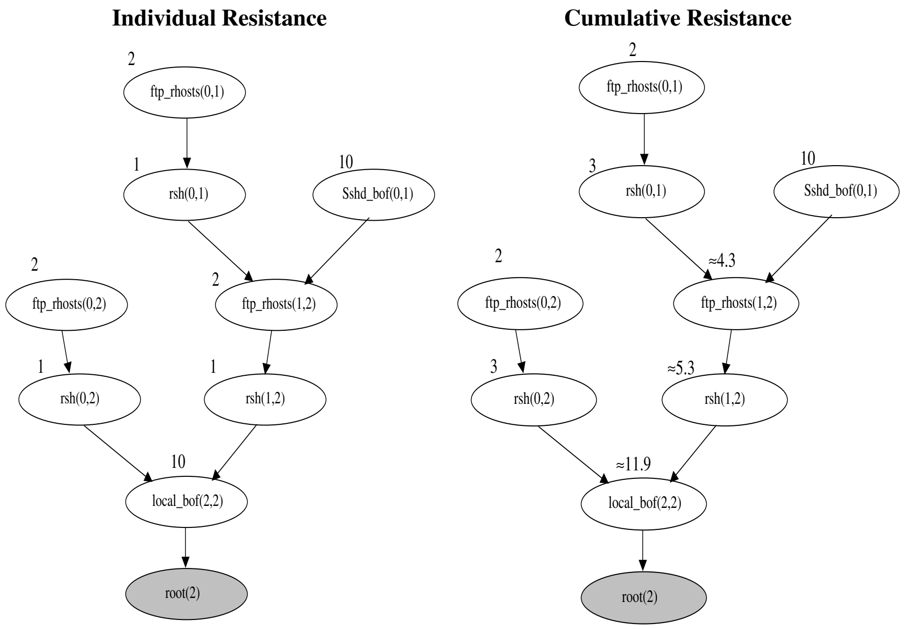


Fig. 2. An Example of Attack Resistance in Real Number

The function $\mathcal{R}$ Next we consider the function $\mathcal{R}$, that is the effect of executed exploits on the individual resistance of other exploits. The previous example is not sufficient for this purpose. Instead, we consider the abstract example given in the left hand side of Figure 3, where the dotted lines represent the following facts. Between exploit 1 and exploit 2, executing one will change the other's individual resistance from the original value x to a new value y. Similar relationships exist between exploit 2 and exploit 3, and between exploit 1 and exploit 6. Notice the special notation between exploits 2, 5, and 7, which denotes the conjunctive relationship between exploits 2 and 5. That is, exploit 7 cannot be executed unless both exploit 2 and 5 are already executed (this may happen when exploit 2 and exploit 5 both imply different conditions, and both conditions are required by the exploit 7).

The left hand side of Figure 3 shows three possibilities in dealing with the function $\mathcal{R}$. First, the effect of $\mathcal{R}(6) = (1, y)$ (that is, executing exploit 6 will change the individual resistance of exploit 1 as $r(1) = y$) can be safely ignored, because exploit 6 can never be executed before executing exploit 1. On the other hand, the individual resistance $r(6)$ can now simply be changed from x to y, because any execution of exploit 6 implies that exploit 1 has already been executed (which in turn implies a change in $r(6)$). Second, there is no naturally induced order between the execution of the exploit 1 and that of exploit 2, so they can be executed in any order. Intuitively, these two exploits *meet* at exploit 6 in the sense that we combine these two resistance values in the same formula when we compute $R(6) = 1/(1/(r(1)+r(4))+1/r(2))+r(6)$. At that point, the last composition operator used is $\oplus$ (that is, the exploit 4 and exploit 2 are

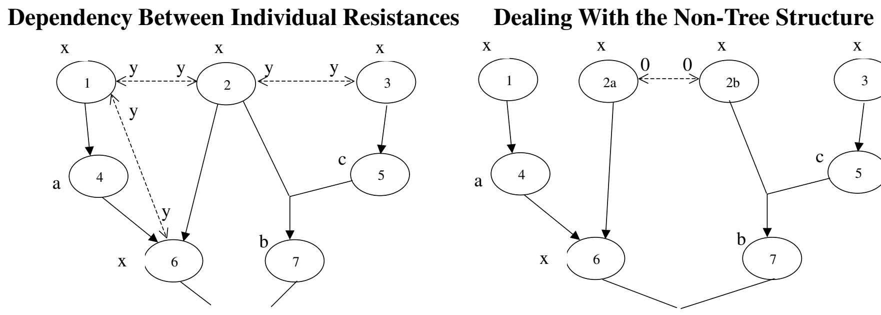

Dependency Between Individual Resistances
Dealing With the Non-Tree Structure
x
y
1
2
2a
2b
3
4
5
6
7
a
b
c
0

Fig. 3. Examples of the Function $\mathcal{R}$

disjunctive). We can then conclude that any minimal attack path (that is, an attack path with no proper subsets being a valid attack path) including exploit 6 will include either exploit 1 or exploit 2, but not both. The effect of $\mathcal{R}(1) = (2, y)$ and $\mathcal{R}(2) = (1, y)$ can thus be ignored.

Third, when exploit 2 and exploit 3 meet at exploit 7 (when we compute $R(7) = r(2) + r(3) + r(5) + r(7)$), the last composition operator we use is $\otimes$. This reflects the fact that the exploits 2 and 3 must both be executed in order to reach exploit 7, although the executions can be in any order. If exploit 2 is executed before exploit 3, then we have $r(2) = x$ and $r(3) = y$; if exploit 3 is first executed, we have $r(3) = x$ and $r(2) = y$. However, we can never have $r(2) = r(3) = y$ because a change only happens after an execution. In this case, we compute the cumulative resistance of exploit 7 for both cases: $r(2) = x$, $r(3) = y$ and $r(3) = x$, $r(2) = y$. We then choose the smaller result as the cumulative resistance of exploit 7. This choice ensures that the computed cumulative resistance will be no greater than the cumulative resistance computed by following any attack path leading to exploit 7. The above discussion covers all possible cases, because when two exploits eventually meet (that is, their resistances are combined), they must meet either at one of themselves (the case of the exploit 1 and exploit 6), or at a different exploit.

The Non-Tree Structure of Attack Graphs. Unlike the nice tree structure in the attack graph in Figure 1, it can be noticed that on the left hand side of Figure 3 both exploit 6 and exploit 7 depend on exploit 2, and the graph is not a tree. This is relevant because the cumulative resistance of this network should be different from another network where exploit 6 and exploit 7 depend on two different exploits. This issue, however, can be easily handled using the function $\mathcal{R}$ as follows. We split exploit 2 into two identical copies, say, exploit 2a and exploit 2b, as shown on the right hand side of Figure 3. We then need the constraint that the resistance of these two exploits will never be added in computing a cumulative resistance, because they actually represent a single exploit. This constraint can be easily modeled as $\mathcal{R}(2a) = (2b, 0)$ and $\mathcal{R}(2b) = (2a, 0)$. We can now compute the metric as usual since the exploit dependency graph becomes a tree.

3.3 Attack Resistance as Sets of Initial Conditions

We consider another concrete case where each exploit's individual attack resistance is the set of initial conditions (that is, conditions not implied by any exploit) required by that exploit. This measure can be easily obtained from the attack graph itself. The attack resistance in terms of the set of initial conditions has a very different meaning from the attack resistance discussed in the previous section. Here the resistance indicates conditions that must be satisfied before an intrusion is possible, instead of the effort and time spent during the actual intrusion. A weakest-adversary metric was recently proposed based on the set of initial conditions [21]. Different network configurations can be ordered based on their relative security, if a subset relationship exists between the sets of initial conditions required for reaching attack goals in the two attack graphs. We show that this metric is equivalent to a special case of our metric by using the set of initial conditions as individual resistance.

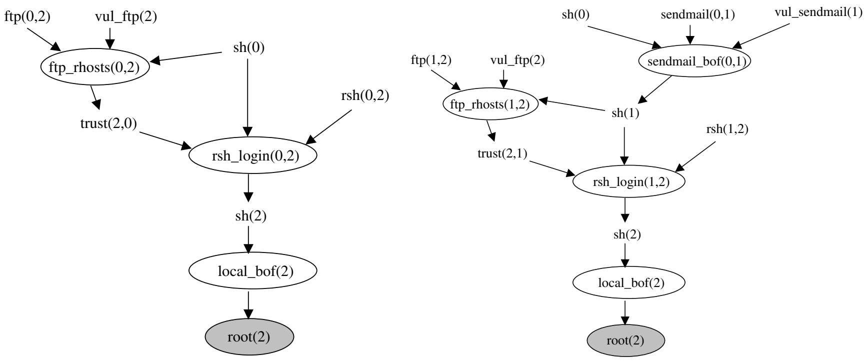


Fig. 4. Two Comparable Network Configurations

Figure 4 shows two network configurations that are comparable based on initial conditions [21]. The left hand side depicts an attack scenario similar to the one in Figure 1 but only involves two hosts. The right hand side shows a different scenario where the attacker is forced by a firewall to exploit the sendmail buffer overflow vulnerability on a third host as an intermediate step. It can be observed that in both cases the goal requires all the exploits to be executed, that is all the dependency relationship is conjunctive. We use set union as the operator $\otimes$ (we do not need the $\oplus$ operator in this case). The cumulative resistance of the exploit $local\_bof(2)$ is thus simply the collection of all initial conditions in both cases. Clearly, the resistance in the first case is a proper subset of the resistance in the second case, and hence the second case has more resistance to potential attacks. This result is the same as reported previously [21].

4 Related Work

An overview of various issues relevant to security metric is recently given in the proceedings of the 2001 Workshop on Information Security System Scoring and Ranking

[2]. The efforts by NIST on standardizing security metric are reflected in the Technology Assessment: Methods for Measuring the Level of Computer Security [18] and more recently in the Security Metric Guide for Information Technology Systems [27], which describes the current state of practice of security metrics, such as that required by the Federal Information Security Management Act (FISMA). Another overview of many aspects of network security metric is given in [10].

Closest to our work, Dacier et. al describe intuitive properties derived from common sense, which should be satisfied by any security metric [7,8,19]. They suggest to assess the difficulty of attacks in terms of time and effort spent by attackers. They assume an exponential distribution for an attacker's success rate over time. Based on this Markov model, they propose to use the MTTF (Mean Time to Failure) to measure the security of a network. They discuss simple cases of combining such measures but do not study the general case. We borrow some of the intuitive properties stated by them, but we use a different way for combining individual measures into the overall attack resistance and we consider a more general case represented by attack graphs.

Our approach of using additional functions for modeling the effect of executed exploits on the resistance value of other exploits is inspired by the work by Balzarotti et. al [3]. However, their work focuses on computing the minimum effort required for executing each exploit, whereas our work computes the overall security of a network with respect to given critical resources. Also, their work does not take into account the kind of dependency that we model using additional functions. Such dependency reduces the difficulty of executing an exploit while not directly enabling it to be exploitable. The work by Pamula et. al introduces a metric based on attack graph [21], in this paper we show that their metric is a special case of ours under certain conditions.

A qualitative measurement of the risk of a network is given based on various forms of the exploitability (that is, whether it is possible to compromise the network) [4]. Another series of work compares software for their relative vulnerabilities to attacks using a fixed set of dimensions, namely, *attack surface* [11,20,13]. The work by Mehta et. al borrows Google's PageRank methodology to rank exploits in an attack graph [14]. Their technique is especially suitable for threat models of worms or other malicious software that spread in a random way in a large network. Our metric has a different threat model, that is attackers have memory and are rational, so in most cases they will not follow a random model.

Metrics for other perspectives of security, especially trust in distributed systems, are relevant to our research. For example, Beth et. al proposed a metric for measuring the trust in an identity established through overlapping chains of certificates [5]. The way they combine values of trust in each certificate into an overall value of trust proves to be useful in our study. Similarly, the design principles given by Reiter et. al are intended for developing metric of trust, but we found these principles applicable to our study as well [24]. The formal logic language introduced for measuring risks in trust delegation in the RT framework inspires us to describe our metric using abstract operators [6].

To obtain attack graphs, topological vulnerability analysis evaluates potential multi-step intrusions based on knowledge about vulnerabilities [7,9,19,22,33,28]. Such analyses can be either forward starting from the initial state [22,28] or backward from the goal state [25,26]. Model checking was first used to analyze whether a given goal state is reachable from the initial state [23,25] and later used to enumerate all possible sequences of attacks between the two states [26]. To avoid the exponential explosion in the number of such explicit attack sequences, a more compact representation of attack graphs was proposed based on the *monotonicity assumption* saying an attacker never needs to relinquish any obtained capability [1]. On the attack response front, attack graphs have been used for the correlation of attacks, the hypotheses of alerts missed by IDSs, and the prediction of possible future attacks [29,30].

5 Conclusion

Presently, qualitative and imprecise arguments are usually the basis for making decisions in securing a network. These arguments can mislead the decision making and as a result cause the reconfigured network to be in fact less secure. This paper described a novel attack graph-based attack resistance metric for measuring the relative security of network configurations. The main components of our metric are two composition operators for computing the cumulative attack resistance from given individual resistances. An additional function allowed the metric to take into consideration the dependency between individual attack resistances. We demonstrated the metric through two concrete cases. First, attack resistance was modeled as a real number, and the case was analogous to computing the resistance of a series-parallel circuit. We showed that the proposed metric satisfied intuitive requirements mentioned in the literature. Second, attack resistance was defined as the set of initial conditions required by each exploit. We showed that our metric in this case resembled the weakest-adversary metric previously proposed. It is our belief that the proposed metric will lead to novel quantitative approaches to vulnerability analysis, network hardening, and attack response.

Acknowledgements. This material is based upon work supported by National Institute of Standards and Technology Computer Security Division; by Homeland Security Advanced Research Projects Agency under the contract FA8750-05-C-0212 administered by the Air Force Research Laboratory/Rome; by Air Force Research Laboratory/Rome under the contract FA8750-06-C-0246; by Army Research Office under grant W911NF-05-1-0374; by Federal Aviation Administration under the contract DTFAWA-04-P-00278/0001; by National Science Foundation under grants CT-0627493, IIS-0242237, and IIS-0430402; and by Natural Sciences and Engineering Research Council of Canada under Discovery Grant. 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. The authors are grateful to the anonymous reviewers for their valuable comments.

References

1. Ammann, P., Wijesekera, D., Kaushik, S.: Scalable, graph-based network vulnerability analysis. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02), pp. 217–224. ACM Press, New York (2002)
2. Applied Computer Security Associates. In: Workshop on Information Security System Scoring and Ranking (2001)

3. Balzarotti, D., Monga, M., Sicari, S.: Assessing the risk of using vulnerable components. In: Proceedings of the 1st Workshop on Quality of Protection (2005)
4. Balzarotti, P., Monga, M., Sicari, S.: Assessing the risk of using vulnerable components. In: Proceedings of the 2nd ACM workshop on Quality of protection, ACM Press, New York (2005)
5. Beth, T., Borcherding, M., Klein, B.: Valuation of trust in open networks. In: Gollmann, D. (ed.) ESORICS 1994. LNCS, vol. 875. pp. 3–18. Springer, Heidelberg (1994)
6. Chapin, P., Skalka, C., Wang, X.S.: Risk assessment in distributed authorization. In: 3rd ACM Workshop on Formal Methods in Security Engineering: From Specifications to Code, ACM Press, New York (2005)
7. Dacier, M.: Towards quantitative evaluation of computer security. Ph.D. Thesis, Institut National Polytechnique de Toulouse (1994)
8. Dacier, M., Deswarte, Y., Kaaniche, M.: Quantitative assessment of operational security: Models and tools. Technical Report 96493 (1996)
9. Farmer, D., Spafford, E.H.: The COPS security checker system. In: USENIX Summer, pp. 165–170 (1990)
10. Hoo, K.S.: Metrics of network security. White Paper (2004)
11. Howard, M., Pincus, J., Wing, J.: Measuring relative attack surfaces. In: Workshop on Advanced Developments in Software and Systems Security (2003)
12. Jajodia, S., Noel, S., O'Berry, B.: Topological analysis of network attack vulnerability. In: Kumar, V., Srivastava, J., Lazarevic, A. (eds.) Managing Cyber Threats: Issues, Approaches and Challenges, Kluwer Academic Publishers, Dordrecht (2003)
13. Manadhata, K., Wing, J.M., Flynn, M.A., McQueen, M.A.: Measuring the attack surfaces of two ftp daemons. In: Quality of Protection Workshop (2006)
14. Mehta, V., Bartzis, C., Zhu, H., Clarke, E.M., Wing, J.M.: Ranking attack graphs. In: Recent Advances in Intrusion Detection (2006)
15. Noel, S., Jajodia, S.: Correlating intrusion events and building attack scenarios through attack graph distance. In: Yew, P.-C., Xue, J. (eds.) ACSAC 2004. LNCS, vol. 3189, Springer, Heidelberg (2004)
16. Noel, S., Jajodia, S., O'Berry, B., Jacobs, M.: Efficient minimum-cost network hardening via exploit dependency grpahs. In: Omondi, A.R., Sedukhin, S. (eds.) ACSAC 2003. LNCS, vol. 2823, Springer, Heidelberg (2003)
17. National Institute of Standards and Technology (Computer Security Division) (2007), http://nvd.nist.gov/
18. National Institute of Standards and Technology. Technology assessment: Methods for measuring the level of computer security. NIST Special Publication, pp. 500-133 (1985)
19. Ortalo, R., Deswarte, Y., Kaaniche, M.: Experimenting with quantitative evaluation tools for monitoring operational security. IEEE Trans. Software Eng. 25(5), 633–650 (1999)
20. Wing, J., Manadhata, P.: Measuring a system's attack surface. Technical Report CMU-CS-04-102 (2004)
21. Pamula, J., Jajodia, S., Ammann, P., Swarup, V.: A weakest-adversary security metric for network configuration security analysis. In: Proceedings of the 2nd ACM workshop on Quality of protection, pp. 31–38. ACM Press, New York (2006)
22. Phillips, C., Swiler, L.: A graph-based system for network-vulnerability analysis. In: Proceedings of the New Security Paradigms Workshop (NSPW'98) (1998)
23. Ramakrishnan, C.R., Sekar, R.: Model-based analysis of configuration vulnerabilities. Journal of Computer Security 10(1/2), 189–209 (2002)
24. Reiter, M.K., Stubblebine, S.G.: Authentication metric analysis and design. ACM Transactions on Information and System Security 2(2), 138–158, 5 (1999)

25. Ritchey, R., Ammann, P.: Using model checking to analyze network vulnerabilities. In: Proceedings of the 2000 IEEE Symposium on Research on Security and Privacy (S&P'00), pp. 156–165. IEEE Computer Society Press, Los Alamitos (2000)
26. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'02), pp. 273–284. IEEE Computer Society Press, Los Alamitos (2002)
27. Swanson, M., Bartol, N., Sabato, J., Hash, J., Graffo, L.: Security metrics guide for information technology systems. NIST Special Publication, pp. 800-855 (2003)
28. Swiler, L., Phillips, C., Ellis, D., Chakerian, S.: Computer attack graph generation tool. In: Proceedings of the DARPA Information Survivability Conference & Exposition II (DISCEX'01) (2001)
29. Wang, L., Liu, A., Jajodia, S.: An efficient and unified approach to correlating, hypothesizing, and predicting intrusion alerts. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 247–266. Springer, Heidelberg (2005)
30. Wang, L., Liu, A., Jajodia, S.: Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. Computer Communications 29(15), 2917–2933 (2006)
31. Wang, L., Noel, S., Jajodia, S.: Minimum-cost network hardening using attack graphs. Computer Communications 29(18), 3812–3824, 11 (2006)
32. Wang, L., Yao, C., Singhal, A., Jajodia, S.: Interactive analysis of attack graphs using relational queries. In: Proceedings of 20th IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec 2006), pp. 119–132 (2006)
33. Zerkle, D., Levitt, K.: Netkuang - a multi-host configuration vulnerability checker. In: Proceedings of the 6th USENIX Unix Security Symposium (USENIX'96) (1996)

Enforcing Honesty in Assured Information Sharing Within a Distributed System

Ryan Layfield, Murat Kantarcioglu, and Bhavani Thuraisingham

The University of Texas at Dallas,
PO Box 830688, Richardson, TX 75083-0688
{layfield,muratk,bhavani.thuraisingham}@utdallas.edu
http://www.utdallas.edu/

Abstract. The growing number of distributed information systems such as the internet has created a need for security in data sharing. When several autonomous parties attempt to share data, there is not necessarily any guarantee that the participants will share data truthfully. In fact, there is often a large incentive to engage in behavior that can sabotage the effectiveness of such a system. We analyze these situations in light of game theory, a mathematical model which permits us to consider behavior and choices for any autonomous party. This paper uses this theory to create a behavior enforcement method that does not need a central management system. We use a simple punishment method that is inherently available in most scenarios. Our approach is applicable to a variety of assured information sharing applications where the members of a coalition have to work together to solve a problem.

1 Introduction

We live in an information-driven world where we use data from multiple information sources to solve problems. The local traffic report, for example, influences choices we make to get to work on time. However, there are several situations where the accuracy of information is crucial to success such as fighting a war or providing medical treatment. We need innovative ways to ensure that the accurate and secure data are shared between the parties. For example, the radio station we get our traffic data from is regulated by the FCC. Creating an honest environment when no central authority is available to enforce behavior presents a new set of challenges. Peers in distributed environments must therefore be sophisticated enough to evaluate the actions of each other and have a consensus of what they should all be doing. The results of these evaluations must be compared with some agreed upon common goal.

In peer to peer systems, a collection of peers is a number of parties that have their own data and want to share data amongst themselves. The goal of this system is to find a way to guarantee everyone ends up with an accurate corpus that reflects all of the data available. Traditionally, this has been done in a hybridized environment of peers and centralized management entities.

Despite the simplicity of their implementation, centralized systems have a number of drawbacks. First, this system must be trusted by all peers to be an impartial judge of activity. Second, such a system by design represents a single point of failure. They can be broken, attacked, and even compromised, effectively eliminating both the usefulness of the system and the services it offers. Third, they must be scaled when more peers want to join the system. In general, centralized management introduces several weaknesses into the data sharing environment. If we want to eliminate the need for such entities, we must find a way for the parties to trust each other.

Parties, however, may have only limited influence over each other. They have entered into this data sharing environment to gather data only because they can offer data. When parties have sufficient motivation to get more out of the system than they give up, the effectiveness of such a system must be questioned. Otherwise, if all parties attempted to cheat the system by sharing bad data in exchange for good data, none of the parties would gain anything.

1.1 Our Approach

Game theory, the study of competition and cooperation through the use of mathematics, offers a solution to this problem. These autonomous parties have no need to share good data since there is no incentive to do so. However, we can assume that they are rational and logical entities that are at least partially interested in collecting good data from the rest of the sources available. We can then assume proper coercion can illicit desirable behavior.

There are several options available in game theory which are designed to elicit behavior. One approach is to create a virtual economy in which data can be bought and sold for a price based on trust. While this is initially a promising solution, it turns out that balancing such an economy without a centralized regulatory entity proves unnecessarily complex [9]. Another option is to fight undesired behavior with undesired behavior: mimic the last data action performed, basically punishing a partner by mimicking their own actions. This works primarily when all data has the same value to all participants, a poor assumption for real-life scenarios. This method is known as Tit-for-Tat, and we discuss it in section 4.2.

We instead explore another alternative: non-participation. Consider a distributed sharing environment in which all of these participants are interested in acquiring each other's data. To keep information private, everyone establishes a secure individual connection with everyone else, forming a fully connected network. Information is traded over these connections simultaneously, where any two parties can exchange pieces of data. In the event that a party believes their trade partner has cheated during an exchange, the link is severed indefinitely. Thus, anyone who chooses to deviate from the desired actions will lose any potential gain from trading with that party in the future.

One of the biggest challenges most applications of game theory face in the real world is the assumption that we know exactly what the other players are doing. Perfect knowledge allows for robust decision making and more efficient choices.

For the sake of realism, we assume that knowledge of party actions comes at a price. Verification is the process of determining what a party has chosen as their actions, and we assign a cost to this process.

Another type of challenge is Trust Management. In a perfect world, everyone is trustworthy and never considers cheating the system or breaching security. The truth is, we deal with distrust due to malicious behavior on a regular basis. The goal of Trust Management is to decide whom to trust and how far we trust them. When we can determine this, dealing with sensitive information becomes much easier. However, as with the issue of perfect knowledge, we rarely know this beforehand. Therefore, a party that wants to properly maintain security must constantly evaluate their peers. In turn, when those evaluations indicate a party is untrustworthy, they must be punished. Punishment allows us to actually force a rational party into becoming more trustworthy by eliminating the benefit of being untrustworthy.

We have two objectives in our work. First, we want to determine the conditions in which non-participation punishment is effective. We use game theory to estimate the existence of these conditions and how they can be made. Secondly, we want to verify the existence of our results by running simulations using a model that simulates changes in behavior.

1.2 Motivating Scenario

Consider the international political environment. We have a number of countries, each with sovereign authority over the affairs of the state and their own set of interests. To protect those interests, we assume that each one has an intelligence agency designed to gather information in the interest of national security. Assume there is some event that, if it is allowed to occur, could threaten the security of any nation to which it happens. The problem these agencies face is that their data are limited. They may have field operatives working in other countries, but in most situations, they deal primarily with the affairs of their own country. If a threat emerged that spanned multiple international borders, it would be difficult for any one agency to track.

Given the severity of the threat, these agencies have decided to establish a mutual agreement: they will share the information they have gathered in exchange for information other agencies hold. Reality dictates that even if they have equal information resources, there is little incentive for agencies to share their real data or to keep the policies associated with it (i.e. secret classification). These organizations tend to reflect the relations that their respective countries have with each other. The agreement makes no provision for requiring any given policy to be enforced due to the lack of a common governing entity. Thus, there is no provision to prevent sabotage within the loose alliance; each agency must invest resources to know what the rest are doing.

Even with a fixed cost of discovering such information, there are a number of factors to consider. If an agency chooses to verify all of the actions taken by others, they will waste resources when their fellow agencies are behaving appropriately. However, if they become too trusting, other agencies can take advantage of this situation without the fear of being caught. We assume that every agency at least has a basic security policy of not sharing bad or corrupted data. Now, consider the use of some punishment and assume that one of these agencies has decided to lie to everyone about recent reports on militia activity. Regardless of the motivation, when that agency is caught, our behavior dictates that they will be isolated from the rest of the data sharing network due to negligent and deviant behavior. Since the information of all agencies are roughly equivalent in value, the loss of this one data source will not affect the ability of the rest of the network to prevent such a militia from causing further trouble. However, when a sufficient number of participants ostracize the offending party, the agency that has been cut off is now left out of what is a valuable group effort. The isolation also serves as a warning to other agencies that may choose to deviate from providing quality information.

1.3 Related Work

Much of our work builds on the foundations of Agrawal et. al. [2]. That paper analyzed the issue of trust management among parties and proposed a solution that uses a management entity that 'taxes parties which use undesirable strategies. This 'tax' comes in the form of a discount on the gain in utility within a game matrix. Our work uses a similar model with two fundamental differences. First, we use a simple withdrawal strategy that terminates the game if bad strategies are used. Second, the responsibility of punishment is completely distributed to all of the parties, eliminating the need for a centralized manager.

In the realm of distributed systems, an area that has garnered considerable attention is that of peer-to-peer file distribution networks. The work here is aimed at enforcing trustworthy behavior in protocols such as BitTorrent [3] and distributed computing. Most of the work we found in this area considers only a third-party, but the works of [10], [5], and [2] deal explicitly with peer-based recourse for deviant behavior.

There is no shortage of game theory driven analysis on behavior enforcement. The work of [1] has inspired our approach to repeated games, but the general works of [5] and [11] have been notable in our efforts as well. None of this research to our knowledge, however, deals with the possibility of refusing to participate within a game. Instead, they suggest choosing a damaging strategy as a form of punishment for a specific amount of time.

Our current research is actually a refinement on our existing work in this field, published as a technical report [8]. We originally attempted to construct a behavior that could govern interaction in a hostile, purely peer-based game that provided the option of either lying or telling the truth. Trade in this case happened between two parties by their own choice, instead of all parties simultaneously. Punishment occurred when certain trade partners were favored more than others, leaving parties that chose to lie with less of a chance of being selected for trade.

1.4 Organization of This paper

Section 2 presents our approach to the game theory, payoff matrix, role of verification, and how malicious behavior is punished. Section 3 is a proof of the sub-optimality of our theory. We discuss how we tested our equilibrium in section 4 along with a detailed listing of significant competing behaviors. The results of our experiments are outlined in 5. Our conclusions, observations, and future directions are left to section 6.

2 Putting a Price on Consequence

We consider our scenario as an application of 2-person evolutionary game theory. The intelligence agencies are represented by game theory agents, which have behaviors and choose a strategy when deciding what to do each round. The measurement of an agency's success is determined by the amount of 'good' data that has been collected. To simplify the array of policies we can choose to enforce, we focus on a simple security policy of telling the truth. Thus, the strategies explicitly available are to Lie, tell the $Truth$, or $Withdraw$, but the option of verification also factors into how an agency can behave.

The value of information varies depending on who receives it and what context they plan to use it in. Data rarely has a uniform benefit to intelligence agencies in the real world. The perception some party i holds about the value of data in a particular round of trade t is Δ_t^i . This value is assumed to be bounded within some range, and represents the raw gain to the party receiving it.

Next, we must address the issue of verification. Using the data acquired during an exchange immediately will obviously cost less than spending resources to verify as long as the information is valid. Therefore, we associate a fixed cost with verification that is uniform among all parties for the sake of fairness. The cost of verification is represented by the constant C_V.

Always verifying results would ensure that the other party never succeeded in deviations such as lying, but it is wasteful with trustworthy parties. Therefore, the probability p_t^i that a single party i will verify the results in a round of transactions t should be inversely proportional to the probability that any given party will tell the truth. Note that no verification allows an attacker to build trust then betray it without consequence.

We assume that agencies have the ability to change their behavior at will. In most real-life situations, parties will periodically change their behavior if they believe it will help them. To accomplish this effect, we adopted the use of a genetic algorithm to allow behavior to 'evolve' among agencies. We save the explanation of this for discussion later.

Based on these observations, we have constructed a payoff matrix that reflects what every rational party should consider. The complete set of actions Γ available to each agency is $[Truth, Lie, Withdraw]$. We assume these actions are only considered on a per-interaction basis; that is, we only consider party strategy choice in pairs during trade.

		Player 1		
		Truth	*Lie*	*Withdraw*
Player 2	*Truth*	$\Delta_t^1 - p_V^2 C_V$ $\Delta_t^2 - p_V^1 C_V$	$\Delta_t^1 - p_V^2 C_V$ $-p_V^1 C_V$	0 0
	Lie	$-p_V^2 C_V$ $\Delta_t^2 - p_V^1 C_V$	$-p_V^2 C_V$ $-p_V^1 C_V$	0 0
	Withdraw	0 0	0 0	0 0

Fig. 1. Payoffs for each pair of strategies during trade

Variable	*Meaning*
Δ_t^i	The value of information offered by agent i in round t of the simulation
p_V^i	The probability that agent i will perform verification
C_V	The cost of performing verification

Fig. 2. The lookup table for variables used in figure 2

The $\{Truth, Truth\}$ strategy is trivial. Both parties expect to receive the utility value of the data from each other, minus the estimated cost of verification should they choose to do so. This is calculated by evaluating how often verification takes place.

Selecting $\{Truth, Lie\}$ or $\{Lie, Truth\}$ is where deviant behavior is introduced. Although we believe equilibrium is virtually impossible to achieve at these points, they must be evaluated: selection of these actions means an equilibrium does not exist yet. Consider two parties i and j that, up to round $k-1$, have been telling each other the truth. Both parties do not expect the other to tell a lie, and as such the probability of verification p_{k-1}^i is at the minimum threshold. It becomes possible therefore in round k that i could lie to j with little chance of being caught. By doing so, i gains the value of j's shared data without ever having to give up significant data of their own, giving i an immediate advantage. If this action is performable without being caught over long periods of time, i can guarantee that they will gain more data and ultimately decrease the effective amounts of information j can acquire.

However, if the choice is $\{Lie, Lie\}$, neither party gains anything. Both parties waste resources for taking the time to interact. Any verification within an equilibrium of this behavior would only add to the loss. In the real-world, this would likely mean that the organization would only benefit if they withdraw from the alliance entirely.

This is the point at which we consider *Withdraw* as an option. When played, the party severs their link with another party, eliminating any further trade. Such an action should be considered a last resort. For example, i has chosen to withdraw from it's connection with j, it will from that point on no longer gain anything from j in future rounds of the game. This would be a tremendous loss, negating any future gains for either party. Therefore, since any strategy choice with $Withdraw$ in it has the same results, $\{Withdraw, Withdraw\}$ becomes an automatic (and undesirable) Nash equilibrium.

Verification has become an interesting factor in the success of behavior. Always choosing to verify would decrease the overall benefit of trade when dealing with a highly reliable source. Reflecting periodic verification checks in the matrix would unnecessarily complicate the game theory and would require a much more complex model.

One of the most interesting characteristics of our application of game theory is the uncertainty of the other party's actions. The nature of the information we consider is not easily verifiable. Most of the research in the area of data sharing does not address games with imperfect information. We believe that reflecting such a property in our work makes our research much more practical, especially when discovering such perfect information comes at a measurable utility cost in the real world.

3 Equilibrium Emergence

Before analyze the above game, we briefly introduce some of the related game theoretic notions.

For any vector $v = (v_1, \ldots, v_n)$, we use v_{-i} to represent $(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots v_n)$, and (v_i, v_{-i}) to denote the reconstruction of the v.

Definition 3.1 ***Nash Equilibrium***[1] *A strategy profile $\sigma^* = (\sigma_1^*, \sigma_2^*)$ is a Nash equilibrium in a two person game with utility functions u_i if the following inequality hold for each agent i,*

$$u_i({\sigma_i}^*, {\sigma_{-i}}^*) \geq u_i(a_i, {\sigma_{-i}}^*)$$

where a_i belongs to set of possible actions A_i that could be taken by agent i

Intuitively, the above definition states that if all agents predict that a particular equilibrium will occur then no player has an incentive to deviate from equilibrium strategy.

Consider a traditional one-shot game. We must pick a strategy in which we can guarantee our success. Consider $Withdraw, Withdraw$ as a natural Nash equilibrium. At first glance, this would appear to be a poor choice. Clearly, better payoffs are found in $Truth, Truth$. However, if we choose $Truth$ as our strategy of choice in this setup, the other player can choose Lie as it increases their utility. If we choose Lie instead, we can take advantage of another player's trust. Should they choose $Truth$ and deviate from the equilibrium, their payoff will dramatically decrease while ours increases; at Lie, our payoff is as we expected. $Withdraw$ of course neutralized both results. Thus, a Nash equilibrium exists at Withdraw,Withdraw.

In practice, not all games are classified as one-shot. Some involve players that play the same game multiple times. Such games enable players to use past data to both predict their opponent's behavior and even affect a particular outcome.

In our model, the "data sharing" game will be played many many times by the participating agents. This scenario can be easily modeled by the "repeated game" ideas from game theory literature [6]. The main observation in repeated games is that the honest behavior in games like the "data sharing" game can be enforced if the game continues to be played with probability $\delta > 0$. In other words, if there are possible future gains, (i.e. if game continues with some probability) each agent can be motivated to be truthful.

We can define the expected payoff for a player i participating in the repeated "data sharing" game as the

$$u_i = (1-\delta)\sum_{t=0}^{\infty} \delta^t \cdot g_i({\sigma_i}^t, {\sigma_{-i}}^t)$$

where $\sigma^t = (\sigma_i^t, \sigma_{-i}^t)$ is the strategy employed at time t, δ is the halting probability of the game, and g_i is the gain achieved at each play of the "data sharing" game. Let $u = (v_1, v_2)$ be the payoff vector of the repeated game. Note that if every period $g_i({\sigma_i}^t, {\sigma_{-i}}^t)$ is equal to some u then u_i will be equal to u.

To illustrate, consider an instance of the game between two intelligence agencies a_1 and a_2 at some point in time on round t. From the perspective of a_1, ${\sigma_{-i}}^t$ is expected to be $Truth$ for a_2 since ${\sigma_{-i}}^{t-1}, {\sigma_{-i}}^{t-2}, \ldots, {\sigma_{-i}}^1$ have all been $Truth$ as well. According to this equation, we should expect the maximum utility of u for $Truth, Truth$. However, a_1 could have a behavior that tries to deviate at round t if a_2 has proven trustworthy. In this instance, σ_i will be Lie, and v will be greater than $Truth, Truth$.

Below we prove that our repeated "data sharing" game can be used to enforce truthful behavior by refusing the deal with dishonest agents that caught cheating. Our proof technique is very similar to the one used for proving "Nash Folk" theorem from the repeated game theory literature [6]. Our main difference as compared to the generic Nash Folk theorem is that in our case opponents actions could not be observed unless a party to choose to verify the correctness of the data. Given the above "data sharing" game, we can prove that truth telling emerges as a Nash equilibrium as follows:

Theorem 3.1 *If telling the truth each round has a gain $g_i > 0$ for both parties then there exits $0 < \delta < 1$ such that telling the truth for both parties is a Nash Equilibrium for "data sharing" game.*

Proof. **Sketch**
We will prove that utility of telling the truth given that the other party tells the truth is bigger then any other strategy that lies with some probability p. To see that let us calculate the expected gain of a given party who chooses to lie with probability $p > 0$ at each round. Note that in a given round with probability $(1-p)$ he will gain $g_{T,T}$ (i.e. the gain achieved when both party tells the truth) and with probability p he will gain $g_{L,T}$ (i.e.the gain achieved when he lies while the other party is telling the truth). If he cheats and is caught, he will earn zero for the rest of the game; otherwise, a new round starts. Under these observations,

we can write the total expected utility of lying with probability p given that the other party verifies the correctness of the received data with probability q as

$$\begin{aligned} u_i &= (1-p)\cdot g_{T,T} + p \cdot g_{L,T} + (1 - p\cdot q)\cdot \delta \cdot u_i & (1)\\ &= \frac{(1-p)\cdot g_{T,T} + p\cdot g_{L,T}}{1-(1-p\cdot q)\cdot \delta} & (2)\end{aligned}$$

Similarly we can write the utility of always telling the truth (denoted as u_i^T below) if the other party tells the truth as

$$\begin{aligned} u_i^T &= g_{T,T} + \delta \cdot u_i^T & (3)\\ &= \frac{g_{T,T}}{1-\delta} & (4)\end{aligned}$$

Note that $u_i^T > u_i$ if we set the δ such that it satisfies the following inequality

$$\delta > \frac{\frac{g_{L,T}}{g_{T,T}}}{\frac{g_{L,T}}{g_{T,T}} - q - 1}$$

Therefore, for the above given δ, telling the truth will be a Nash equilibrium because each party has no incentive to lie given that the other party is telling the truth.

4 Simulation Construction

Obviously, if every party used the game theory we proposed as their primary logic, we would have no issue with reaching an equilibrium immediately. However, we would instead like to see how our design interacts in a variety of game environments. A diverse environment will enhance the robustness of our theory.

The gaming environment of our design is straightforward. We have a collection of N game theory agents representing parties that interact via secure bidirectional communication pipes. Each party a_i is initially linked to every other party in the system, forming a fully connected graph. This pipe can be broken voluntarily by the party at either end. We assume this pipe is completely secure from tampering or eavesdropping for the sake of simplicity. All players act simultaneously in each round.

We wanted to analyze the results of our theoretical conclusions in a diverse environment of party behaviors. In order to do this, we use three existing possible approaches to this scenario (*Random*, *Tit-For-Tat*, *Dishonest*), our own behavior *Truthful-Punisher* along with two variations on our own work (*Liar*, *SubtleLiar*).

4.1 Random Behavior

The *Random* behaviors simply randomly selects *Lie* or *Truth*. This strategy represents a party which has no desire to spend time on the details of the alliance while simultaneously lacking a consistent motivation to adopt proper behavior. In theory, this randomized behavior can succeed when other parties do not consider the past and there is little effect due to punishment.

4.2 Tit-for-Tat Behavior

Next, we have the famous *Tit-For-Tat* strategy. A party using this strategy starts by telling the truth. After that, this party mimics whatever action was taken by their trade partner. Research has proven that, unless other parties conspire against it in some fashion, this is the most effective behavior possible for games resembling the Prisoner's Dilemma, as discovered by Anatol Rapoport [4].

Within our scenario, this behavior operates at a disadvantage. Since perfect information is not free, the party must verify the results of each and every trade they make. This could lead to a situation in which it actually gains less utility against behaviors that are relatively trustworthy but have little regard for verification.

However, it also has a potentially larger advantage: it does not use the grim trigger punishment system. The idea behind punishment is that the party takes a calculated "hit" to the immediate trade benefits by refusing to deal with parties that do not tell the truth, in the hopes that they will become more honest. While this has obvious ramifications for dishonest behavior, unless interacting with that party has a net loss (i.e. tells a lie more often than it tells the truth), it is still beneficial to maintain an open communications channel and choose the less harsh strategy of mimicking their choice. In essence, this behavior should provide the best competition to our own construction.

4.3 Dishonest

In order to add the appropriate amount of realism to our scenario, we must also consider parties that have no desire to contribute meaningfully to the group. Such behaviors are simply classified as *Dishonest*, and as such they choose to always lie. They still may reap the benefits of those that choose to tell them the truth, but they will never bother to verify what they receive nor punish those that lie as well. Thus, this agent exists in our simulation solely to insure the rest of the parties cannot make the assumption that all behaviors will ultimately yield any sort of positive or 'break-even' net gain. This is in contrast with the *Random* behavior, which will arguably still yield a net gain of zero through prolonged participation.

4.4 Truthful-Punisher Behavior

Before we describe the variations on our ideal behavior, we must first describe what our game theory analysis has suggested to us. Since there is a clear Nash Equilibrium at $\{Truth, Truth\}$ with our punishment modifications, our behavior always chooses $Truth$. The probability of verification is done as a percentage that is handed off as part of the behavior characteristics. When the simulation is first created, each time a party using this behavior type (or a variant) is instantiated, a random percentage is chosen for its' verification probability. Essentially, this party either tells the truth or cuts the other party off.

4.5 Periodic Liar Behavior

The first variant on our behavior is to try and get away with lying a fraction of the time. This is designed to represent an party that believes they can deviate from time to time when they have a desire to sabotage their competition. More importantly, it simply represents a mindset in which the party does not believe that the original conclusions of always telling the truth is a true equilibrium within the 'real-world' environment, making it a close relative of the *Random* behavior.

4.6 Subtle Liar Behavior

In theory, any party could choose to deviate only when they know that their trade partner is going to give them valuable data. They believe they can lie without worry of significant punishment. We assume that party i will choose $\{Lie\}$ during communication with j whenever $\Delta_t^i > \Delta_T$ during round t, where Δ_T is simply a threshold above a significant majority of all possible piece values. This is especially handy when dealing with Tit-For-Tat, as retaliatory behavior assumes that by lying to them on the next round will neutralize gains from deviation. Since piece values vary over a set range, this works to the behavior's advantage as long as the data it will not receive due to punishment during the next round is of lesser value.

5 Experiments

Information is exchanged between our virtual parties every round. During each round, a party trades with the rest of the parties they are connected to. No one party has an advantage over the other through knowledge of the move their partner has made due to the synchronous nature of our setup.

All experiments are run to no more than 20,000 rounds. The game will terminate early if an equilibrium is achieved (i.e. all agencies go to a particular behavior, leaving no other possible behaviors to choose from). Note that we are not explicitly using game theory approaches for infinitely repeated games; we assume that at some point there will no longer be a need for the alliance.

We judge a party's fitness by the value of accumulated data. Each time a party is told the truth, the data value is added to the total value of the party. Whenever a party chooses to verify, the cost of verification is subtracted. Obviously, being told a lie and choosing to verify the data will result in a net loss. In the spirit of our scenario, a positive gain in data is much more desirable.

Every L rounds, we pause the game to perform an evaluation of the performance of these parties. This reflects when agencies choose to evaluate their performance to maximize efficiency. First, we need to see how each of them has performed since the last check. Every party's gain q_i is calculated from the increase in their net value. This value is added up to yield a total utility value over the whole system, q_T.

Next, we calculate $p^i_{behavior} = q_i/g_T$ for each party's behavior, where $p^i_{behavior}$ denotes the normalized probability that the behavior held by party i should be used by parties in the next generation. A behavior embodies both the core logic mentioned in section 4 along with any attributes. We use this normalized percentage as a way of measuring how well a particular approach has performed in contrast with the rest of the system. The higher a party's relative gain, the higher their percentage 'score'.

We want our population to reflect the fitness of each behavior proportionally, according to the basics of genetic algorithms[12]. In our scenario, we assume that agencies will want to maximize their data trading success by adopting the behaviors of those that are most successful. Since $\sum_{i=0}^{n} p^i_{behavior} = 1$, we can use $p^i_{behavior}$ to ensure that the next 'generation' of agencies adhere to this principle. We thus reassign the strategies for every agency based on this probability. We do not consider the very real notion that a successful agency is unlikely to want to change it's strategy; we simply need the population to reflect the evolved characteristics of the system.

We expect a number of properties to emerge based on our analysis. First, we expect that our equilibrium-based behavior to outperform and ultimately dominate the overall population given enough generations. Next, we expect that this behavior will adjust it's verification rates based on how many parties use a deviant behavior. Finally, we expect our approach to dominate all possible variations available.

5.1 Results

The outcome of our experiments confirms our theory is correct. Verification and punishment appear to be highly effective even in a diverse population, as our simulation consistently converged to a homogeneous population of our particular behavior. There's a clear correlation between the use of our punishment method and the success of agents within the system. Since agencies that did not obey the truth policy were cut-off, agencies which told the truth remained within a somewhat exclusive clique. As long as this clique's benefits exceeded those that are offered by those outside of it, the system eventually began to encompass only agencies that used the adhered to an honest strategy remained after just a few rounds.

As we suspected, *Tit-For-Tat* did not perform well enough to beat our strategy. Despite the fact that the strategy kept links open to several parties which still offered a net gain, the population eventually became devoid of any dishonest participants. Once this happened, *Truthful-Punisher*'s ability to settle for less than perfect information (via infrequent verifications) gave it a clear advantage, as fewer resources were wasted verifying information that would never be a lie. It at times took several generations, but eventually, *Tit-For-Tat* would disappear from our population, leaving only *Truthful-Punisher* with complete dominance.

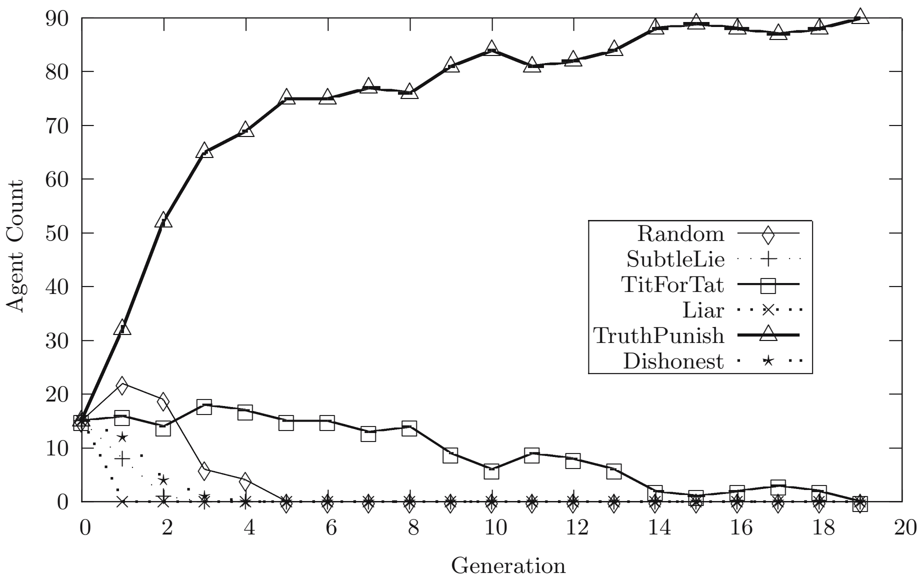


Fig. 3. Behaviors by population per generation

The rest of the behaviors which choose not to tell the truth consistently did not perform well enough to pose a threat to our equilibrium. Despite the two variants which attempted to lie only periodically, those that choose such an approach were eventually caught and collectively punished. The only time this did not happen was when we set a very low threshold for the SubtleLiar behavior; when the probability of lying was less than 10%, the difference made to the agency's performance was so low that our experiments converged to SubtleLie with equal probability. However, since the net gain from such periodic lying was also very low, we suspect that this has more to do with a small fraction of difference lost in the varying cost of the pieces. Additionally, the probability that they would be caught was simply too low to be of significant value. At rates at or above this threshold, our original behavior prevailed consistently against it.

Convergence to our behavior happened in an average of 20 generations. The leading competitor often ended up clinging to a small portion of the population as a handful of agents before eventually succumbing to the agents bearing our behavior. Typically, this was a small sub-population of no more than 5 parties, which were usually *Tit-For-Tat*. We believe the reason for this is rooted in the fact that our own constructed behavior tends to seek a verification probability based on how honest the population is at a given time. As our behavior propagates, the need to verify decreases, leaving it more vulnerable to future attacks that never occur.

One of the ways in which we observed system performance is by way of four metrics: *Truces*, *Fools*, *Follies*, and *DeadLinks*. *Truces* represent both parties choosing to tell the truth. *Fools* are situations in which one party told the truth

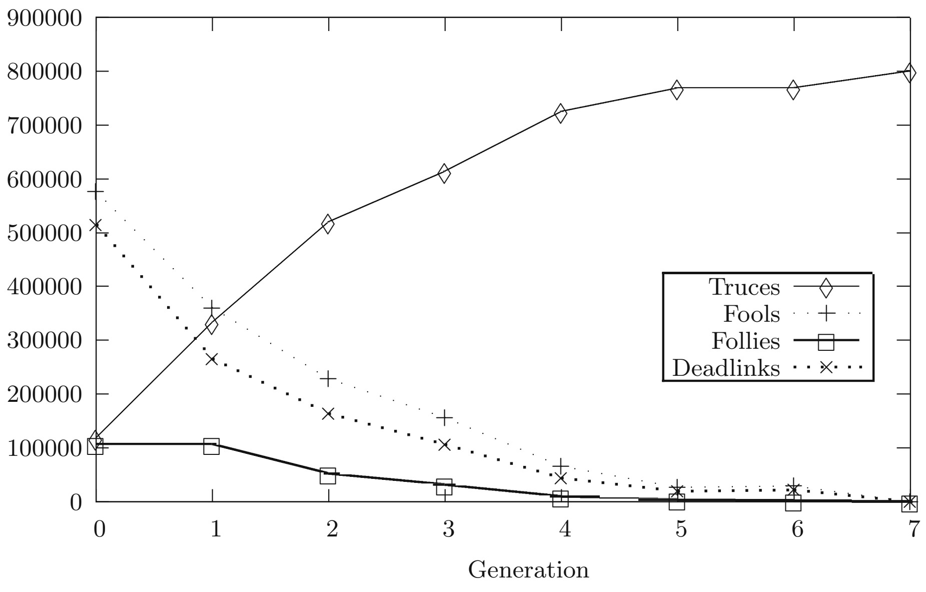


Fig. 4. Chosen strategy pairs per generation

while the other lied. *Follies* represent when both parties choose to lie to each other, resulting in either no gain or net loss. *DeadLinks* simply indicated when trade never happened between two parties. In Figure 4, we see that *Truces* exceed the other options in only four generations, indicating that the system is achieving optimality as early as possible.

The dynamic behavioral choices of any given iteration of our simulation involve the life and death of the various behavioral choices, as seen in Figure 3. Starting from an equal distribution of six different behaviors, we find that the first variant *Liar* 'dies' after only one generation. The next death is of the *SubtleLiar* variant in the generation following. Both of these variants are surprisingly beaten by the *Random* behavior; clearly, even small deviations from our proposed behavior can prove disastrous. Finally, and least surprisingly, is the delayed demise of *Tit-For-Tat*, which is the last behavior to go. This is most likely due to the fact that it will never lie to a trustworthy opponent which punishes; thus, it is never isolated from the productive members of the group and only loses due to dependency on perfect information.

6 Conclusions

Overall, we were pleased with the results of our simulations. The populations converged rather quickly to a $\{Truth, Truth\}$ equilibrium, and our behavior eventually overcame any competition provided there was enough time. Dishonest behaviors were eliminated rather quickly, even in variants of the experiments we performed with unfair advantages given to competing behaviors. Although their were restrictions in involved on its' effectiveness, we were overall pleased to say that our work has fulfilled our objectives.

Despite this success, we were not entirely satisfied with the results. Clearly, there is no need to continue verifying results once the system converged to a Truth strategy. We originally asserted that the verification rate would thus go to our lower bound for verification (10%), as agents using a strategy reflecting little or no verification should pull ahead. Instead, our system simply approached a 30% rate with significant deviation. The problem we believe lies in the nondeterministic nature of our choice of genetic algorithms. Since verification costs are relatively small compared to the payoff from the information, there is always a net gain regardless of verifying the information when the truth is always told. However, even when we doubled the cost of verification and set payoff in $\{Truth, Truth\}$ to be a constant, there was simply never enough of a gain to converge.

The last question we want to answer is how effective our agent is as a group. Obviously, a single agency cutting off others is not going to be a significant deterrent on their own. Our results show that approximately 40% of the population must use punishment to significantly deter others from deviating from $Truth$. This reflects similar findings found in distributed computing, such as the Byzantine Generals Problem, in which a certain majority of the participants must be trustworthy in order to properly succeed against deviant strategies [7].

Another pressing issue is the vulnerability of the population to constructed behaviors which could wait for convergence to a mostly honest population and then switch to a dishonest policy of strategy choice. Since the characteristics of our design favors a more vulnerable state when it appears 'safe' to do so, we are concerned that future generations would have little defense against a growing dishonest population. The only way we could combat this is to introduce mutation rates among our behavioral characteristics.

The most endearing application of our work is how it can apply to the enforcement of any desired behavior. The nature of the Folk theorem is that, with sufficient patience and time, any desirable equilibrium can be achieved. Based on this work, we can enforce any security policy as long as the actions taken are verifiable in some capacity. Given the traditional approaches that require a management party, true distribution of responsibility makes it possible to have much more robust security that does not rely on any one entity to enforce behavior.

In order to bring more realism to our model, our future work will also address two major assumptions: imperfect verification and insecure lines of communication. The former deals with situations in which we cannot guarantee information will be properly classified as a truth or a lie. The latter raises the possibility that we have insecure communication channels; a would-be attacker could easily cause communication disruption through information tampering. Both can be addressed by assigning a confidence factor in the form of a probability reflecting the likelihood that data can be trusted while relaxing the grounds on which the $Withdraw$ option is selected.

References

1. Agarwal, N.: Equilibrium Game Theory Under the Conditions of Repeatability. SSRN eLibrary (2002)
2. Agrawal, R., Terzi, E.: On honesty in sovereign information sharing. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Boehm, K., Kemper, A., Grust, T., Boehm, C. (eds.) EDBT 2006. LNCS, vol. 3896, Springer, Heidelberg (2006)
3. Andrade, N., Mowbray, M., Lima, A., Wagner, G., Ripeanu, M.: Influences on cooperation in bittorrent communities. In: P2PECON '05. Proceeding of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems, New York, NY, USA, pp. 111–115. ACM Press (2005)
4. Axelrod, R.: The Evolution of Cooperation. Basic Books (1985)
5. Buragohain, C., Agrawal, D., Suri, S.: A game theoretic framework for incentives in p2p systems. In: P2P '03. Proceedings of the 3rd International Conference on Peer-to-Peer Computing, Washington, DC, USA, p. 48. IEEE Computer Society (2003)
6. Fudenberg, D., Tirole, J.: Game Theory. MIT Press, Cambridge, Mass (1991)
7. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. 4(3), 382–401 (1982)
8. Layfield, R., Kantarcioglu, M., Thuraisingham, B.: Research and simulation of game theoretical techniques for data sharing among semi-trustworthy partners. Technical Report UTDCS-46-06, The University of Texas at Dallas (2006)
9. Mahajan, R., Rodrig, M., Wetherall, D., Zahorjan, J.: Experiences applying game theory to system design. In: PINS '04. Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems, pp. 183–190. ACM Press, New York, NY, USA (2004)
10. Monderer, D., Tennenholtz, M.: Distributed games: from mechanisms to protocols. In: AAAI '99/IAAI '99. Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence, Menlo Park, CA, USA, pp. 32–37. American Association for Artificial Intelligence (1999)
11. Myerson, R.: Game Theory: Analysis of Conflict. Harvard University Press, Cambridge, Mass (1991)
12. Riolo, R., Worzel, B.: Genetic Prgoramming Theory and Practice. Kluwer Academic, Boston (2003)

A Privacy-Enhanced Attribute-Based Access Control System

Jan Kolter, Rolf Schillinger, and Günther Pernul

Department of Information Systems, University of Regensburg,
D-93040 Regensburg, Germany
{jan.kolter,rolf.schillinger,guenther.pernul}@wiwi.uni-regensburg.de

Abstract. Service-oriented architectures (SOAs) are increasingly gaining popularity due to their considerable flexibility and scalability in open IT-environments. Along with their rising acceptance comes the need for well suited security components. In this respect, access control and privacy emerged to crucial factors.

Targeting the demands of a SOA, many promising authorization models have been developed, most notably the attribute-based access control (ABAC) model. In this paper we take up concepts from the OASIS XACML and WS-XACML specifications and introduce a dynamic ABAC system that incorporates privacy preferences of the service requestor in the access control process. Separating the Policy Decision Point from the service provider's premises, our infrastructure enables the deployment of alternative PDPs the service requestor can choose from. We employ a PKI to reflect the sufficient trust relation between the service provider and a potential PDP. Our work is carried out within the European research project Access-eGov that aims at a European-wide e-Government service platform.

1 Introduction

The trend towards large distributed IT-systems is ongoing and promises many economical and technical advantages. Especially the service-oriented architecture (SOA) paradigm [1] emerged to a popular and widely adopted technology, as SOAs allow the easy wrapping of existing distributed applications into platform independent web services. Distributed architectures and SOAs in particular bear large potential for both service providers and service consumers. For service providers the main incentives of a SOA lie in the low maintenance of large service infrastructures. Furthermore, SOAs facilitate the bundling of single, physically divided services. With a flexible SOA a service provider can easily outsource selected links of its value chain to specialized providers. Service consumers, on the other hand, profit from easy access to services as well as standardized discovery and execution of web services.

The distributed and flexible character of SOAs calls for well fitted security mechanisms in place. Especially access control and privacy are security concerns that represent decisive factors for the building of a secure and trustworthy service infrastructure. Access control has been addressed frequently in the context of SOAs. Targeting the flexible and dynamic requirements of SOAs the attribute-based access control (ABAC) model is considered an appropriate approach for an access control system [2]. Recent standardization efforts of the OASIS have created the XACML specification [3], a standard suitable for the implementation of an ABAC system. The WS-XACML working draft [4] specifies this standard for the use in a SOA environment.

Apart from flexible access control, privacy increasingly moves to the center of attention in distributed IT infrastructures. Respective studies show that users become more and more concerned about the disclosure of private attributes [5]. In the ABAC authorization model, however, the disclosure of personal attributes (subject attributes) is essential for the access decision. Consequently, privacy plays a special role in distributed architectures that employ an ABAC system.

In this paper we introduce an ABAC system that incorporates privacy-preserving components. We enable the service requestor to define individual attribute disclosure rules. These privacy preferences are utilized to select one out of many alternative Policy Decision Points (PDPs). As we decouple the PDPs from the service provider's premises, it becomes more likely that the characteristics of one PDP match the service requestor's attribute disclosure rules. Introducing a PKI in our infrastructure, we address the fact that the service provider must fully trust the entitled PDPs in their unbiased decision making capabilities.

The content presented in this paper has been developed within the European research project Access-eGov[1]. The project's goal is to develop a European-wide e-Government platform that facilitates the semantic discovery of individual and bundled services. A cornerstone of this platform is a powerful security infrastructure that provides dynamic authorization functionality for public authorities and protects the citizen's privacy at the same time.

The remainder of this paper is organized as follows: Section 2 discusses relevant ABAC approaches and selected privacy technologies. In Section 3 we present the concept and implementation details of our privacy-enhanced ABAC infrastructure. Section 4 lays out details about the integration into the research project Access-eGov. After describing related work in Section 5, Section 6 finally concludes the paper and gives an outlook on planned future work.

2 Fundamentals

In this section we describe basic access control and privacy concepts and technologies. Based on these inputs we present our privacy-enhanced security architecture in Section 3.

2.1 Access Control

Access control is generally defined as the prevention of unauthorized use of a resource, which includes the prevention of use of a resource in an unauthorized

[1] European Union, IST Program, Contract No. FP6-2004-27020.

manner [6]. As confidentiality is a basic requirement for every interaction, access control is considered an important security functionality. Over time various authorization models have been developed, most notably the Discretionary Access Control (DAC) model, the Mandatory Access Control (MAC) model and the Role-based Access control (RBAC) model [6]. However, all of these approaches do not meet the requirements of large-scale distributed environments like SOAs (see Section 1). Here, an access control system is needed, which is flexible enough to handle the inherent heterogeneity of users.

Addressing the needs of distributed systems, the attribute-based access control (ABAC) model evolved. Due to its flexibility and its capability to express complex access control semantics ABAC is deemed a suitable authorization model for SOAs [2]. The ABAC model moves away from the static definition of access permissions and is based on the comparison of subject and object descriptors, allowing for dynamically grouping objects and subjects [7]. Unlike an RBAC approach, this grouping is not done manually by a system administrator but implicitly by subject and attribute values. A big advantage of ABAC becomes evident, if one considers an access policy that grants access to a certain resource depending on the service requestor's age. As an attribute like the age changes continually, the access policy needs to be evaluated dynamically.

The XACML specification [3] is an OASIS standard that supports the integration of subject and object attributes in access policies, a feature that is essential for ABAC policies. The standard defines a powerful policy language that supports complex, fine-grained rules. Rules are aggregated to policies that control the access to a resource. For a detailed review of the XACML policy elements the reader is referred to [3].

Along with the policy language the XACML standard defines an authorization infrastructure that is generic enough to implement the ABAC authorization model [8]. Fulfilling the needs of distributed architectures, the XACML architecture logically separates the access control components responsible for policy definition, policy enforcement and policy evaluation. Specifically, the XACML architecture specifies the implementation of a Policy Enforcement Point (PEP), a Policy Administration Point (PAP), a Policy Decision Point (PDP), a Policy Information Point (PIP), and a Context Handler. Each of these actors is devoted to one specific task of the access control process: The PEP receives access requests and forwards them to the PDP which is responsible for the evaluation of attributes and the access decision. The PIP supplies the PDP with subject and object attributes that are relevant for the access decision. The access policy is provided by the PAP that stores and maintains the access rules. The XACML architecture also employs a Context Handler to collect and broker data flows.

The Web Service Profile of XACML (WS-XACML) specification [4] defines means for the application of XACML in a SOA. While WS-Security [9] introduce security tokens in the context of web services, WS-XACML specifies the Authorization Token, a format that allows the transfer of the access decision to a trusted third party. This enables the trusted third party to make an access decision on behalf of the service provider. Furthermore, WS-XACML defines a format for the expression of authorization, access control, and privacy policies of web services, areas that are not covered by the common W3C standard WS-Policy [10]. By means of policy assertions, WS-XACML facilitates the definition of requirements and capabilities with respect to authorization and privacy on client and service side. Matching semantics regulate that the capabilities of the service provider must match the requirements of the client, and vice versa. Additionally, WS-XACML defines the relation of P3P policy preferences and XACML policy assertions.

2.2 Privacy

User behavior and the way users disclose personal data have changed significantly over time. A main reason for this development is the growth of the Internet to a multi-million user platform that is used for various needs and wants. Especially eCommerce and trends like interactivity of web sites and personalized offers strongly rely on personal user data.

An increasing number of users, however, perceive this trend as a privacy threat, as they need to disclose more and more personal information to a growing number of providers [5]. Addressing these concerns the Platform for Privacy Preferences (P3P) [11] provides a privacy policy language which enables service providers to advertise their individual privacy policy. A P3P privacy policy describes how personal data of users are dealt with, including information about the purpose and the recipients of the collected data. On client side, privacy preferences of the user are collected and translated into "A P3P Preference Exchange Language" (APPEL) [12]. A privacy agent then uses this information to signal if a web site's privacy policy is in line with the user's pre-defined privacy preferences.

3 A Privacy-Enhanced ABAC System

In this section we lay out the conceptual and technical aspects of an ABAC-based access control infrastructure that incorporates privacy disclosure rules of the client. The section starts with a brief outline and a description of our goal.

3.1 Outline and Goal

As described in Section 2, the basic idea of an ABAC system is to grant access to a resource based on static or dynamic attributes of the service requestor (subject) and the resource itself. The main advantage of ABAC lies in its dynamic character which makes the access control process flexible and satisfies the needs of SOAs. However, ABAC strongly relies on the disclosure of privacy-sensitive subject attributes service requestors are not willing to share in any circumstance. Especially users of large-scale IT-infrastructures have a rising interest in the controlled disclosure of their attributes. Addressing these concerns of users, we introduce a security infrastructure that enhances the ABAC system specified by the XACML specification [3] with privacy preserving components.

In order to enable the client to control the transfer of privacy-sensitive attributes, we propose to take up common ideas from privacy enhancing technology (PET) approaches. As a manual approval by the client for each service access seems obviously unrealistic, clients should define their individual privacy preferences (attribute disclosure rules) [11,12]. These rules should specify what attributes a client is willing to disclose under which circumstances. When accessing a resource protected by an ABAC system, these preferences should be considered dynamically when the system determines the set of attributes a client needs to provide (see example in Section 3.2).

With clients defining their individual privacy preferences, it is not unlikely that attribute disclosure rules forbid the transfer of certain attributes to a particular service provider. For this reason, we utilize the flexible character of the XACML architecture (see Section 2) and separate the Policy Decision Point (PDP) from the service provider's premises [13]. In this scenario a direct transfer of client attributes to the service provider itself is not necessary, as the outsourced PDP is the only actor that collects and evaluates attributes.

The separation of the PDP from the service provider even facilitates the deployment of many different PDPs a client can choose from. Each individual PDP offers a variety of capabilities. In a dynamic selection process a PDP is chosen whose capabilities match a client's privacy preferences. Consequently, the access decision is made by a PDP the client trusts and is comfortable to share certain attributes with. This scenario makes it more likely for a client to find a suitable PDP that does not conflict with personal attribute disclosure rules.

At this point it is noteworthy that a breakup of the PDP from the service provider's access control infrastructure is only possible, if the service provider fully trusts the evaluation and decision making processes of the PDP. The proposed infrastructure acknowledges this aspect and incorporates a Public Key Infrastructure (PKI) that is used to reflect the service provider's level of trust in a PDP. In the following the described infrastructure is presented in detail starting with a conceptual view of the architecture.

3.2 Architecture

The ABAC system sketched in the last section is built on the XACML architecture (see Section 2). In the XACML architecture the access control process is split between several actors, namely a Policy Enforcement Point (PEP), a Policy Decision Point (PDP), a Policy Administration Point (PAP) and a Policy Information Point (PIP).

In order to dynamically choose a PDP that matches a client's individual privacy preferences, we extend the XACML architecture with elements and actors that contribute to privacy and trust in the dynamic access control process. Figure 1 depicts our proposed architecture.

Apart from the common access control actors, the figure shows a set of alternative PDPs that are capable of performing the access decision. The PDP Selector on client side is responsible for the browsing, matching, and the selection of a proper PDP.

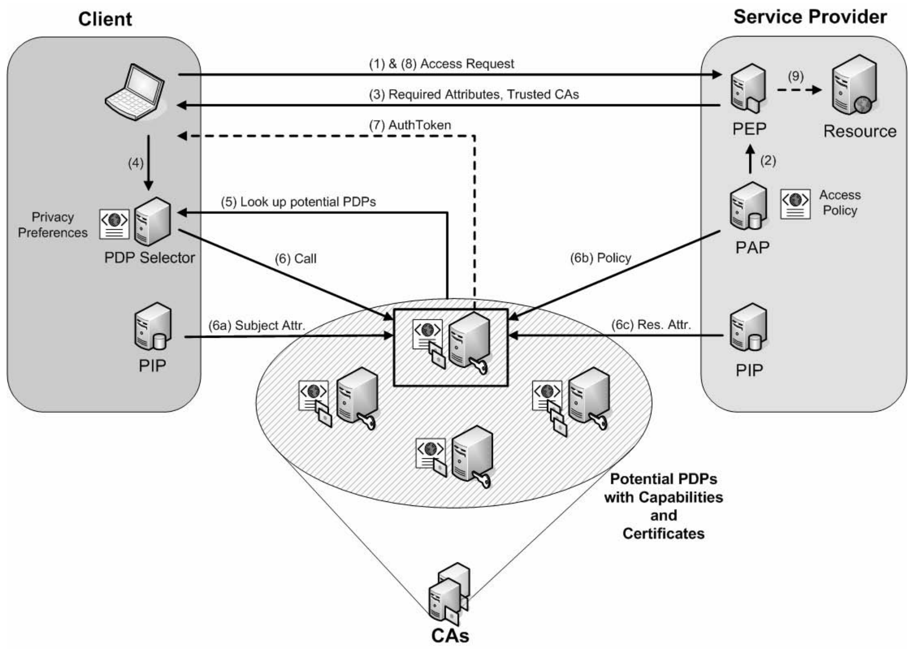


Fig. 1. An ABAC system enabling the dynamic selection of a privacy-conform PDP

The fact that the separated PDPs are entitled to make the access decision outside the influence of the service provider requires a maximum level of trust of the service provider, as an outsourced PDP is solely responsible for the evaluation of the required attributes and the processing of the access policy. Employing a PKI, we introduce the idea of trust certificates being held by each PDP. These certificates reflect the trust of service providers in the evaluation capabilities of a PDP. Specifically, they state that a certain PDP is authorized to evaluate a certain attribute. As not every service provider will trust every Certificate Authority (CA), each PDP can hold attribute evaluation certificates from several CAs for a particular attribute.

Focusing on the sequential steps of the access control process, our scenario starts with a client accessing a protected resource of the service provider (1). The responsible PEP receives the access request and calls the PAP that holds the access policy of the resource (2). The PAP then reads the access policy and determines the subject attribute set required for an access decision, which is subsequently forwarded to the client (3), e.g. a credit card number and the date of birth. At this point, the client is also supplied with the CAs the service provider trusts.

We point out that the required attributes can also include alternative attribute sets. Increasing the chance to get access to the requested resource, the client can opt to transfer several alternative attribute sets. However, in the process of finding a proper PDP it is likely that only the transfer of a few alternative attribute sets will be in accordance with the client's privacy preferences.

After the required attributes have been transmitted, the client calls the PDP Selector (4) to browse for potential PDPs (5). In a first step, the PDP Selector filters all PDPs that do not hold a certificate for the required attributes which need to be evaluated. Certificates must be issued by a CA the service provider trusts. The PDP Selector is provided with that information in step (2) and (3).

As mentioned before, a suitable PDP must also meet the privacy preferences of the client. Such requirements for example limit the disclosure of a credit card number to financial institutions. They could also define that the date of birth is only transferred, if a secure connection has been established. For this reason, each potential PDP advertises a set of capabilities that define service attributes and technical aspects. After the PDP Selector has looked up and filtered available PDPs, the capabilities of the remaining PDPs are matched against the client's privacy preferences. If for example the credit card number and the date of birth are required subject attributes, only PDPs under the control of a financial institution that can establish a secure connection are actors the client is comfortable to transfer the required attributes to.

Once a suitable PDP has been selected, the PDP is called with the service request (6) and the required subject attributes from a client side PIP (6a). It also receives the access policy (6b) and required resource attributes (6c) provided by the PAP and the PIP of the service provider. Subsequently, the PDP evaluates all required attributes, processes the access policy and makes an access decision. In case of a positive access decision ("Permit"), the PDP issues an authorization token to the client (7). The client in turn uses the authorization token to access the resource (8). The token is acknowledged by the PEP which grants access to the resource (9).

Figure 2 shows a sequence diagram of the presented access control process, which points out the role of each participating actor in the access control architecture. The following section focuses on technical aspects and standards we employed to develop the proposed infrastructure.

3.3 Technical Details

Employing the proper technical means for the dynamic selection of a PDP is vital to the success of our envisioned infrastructure. Section 2.1 and Section 2.2 already presented specific policy languages capable of describing privacy preferences. P3P is a notable representative, as its advantage lies in the most important aspect of privacy descriptions, the user-friendliness. P3P, however, is not suitable to describe the actual application of privacy preferences. This task rather requires policy languages like XACML. For this reason, P3P and XACML are considered as complimentary technologies [14], with P3P describing the preferences on a rather high level and XACML allowing an "instantiation" of a P3P policy, describing it at a lower level which involves specific attributes and rules.

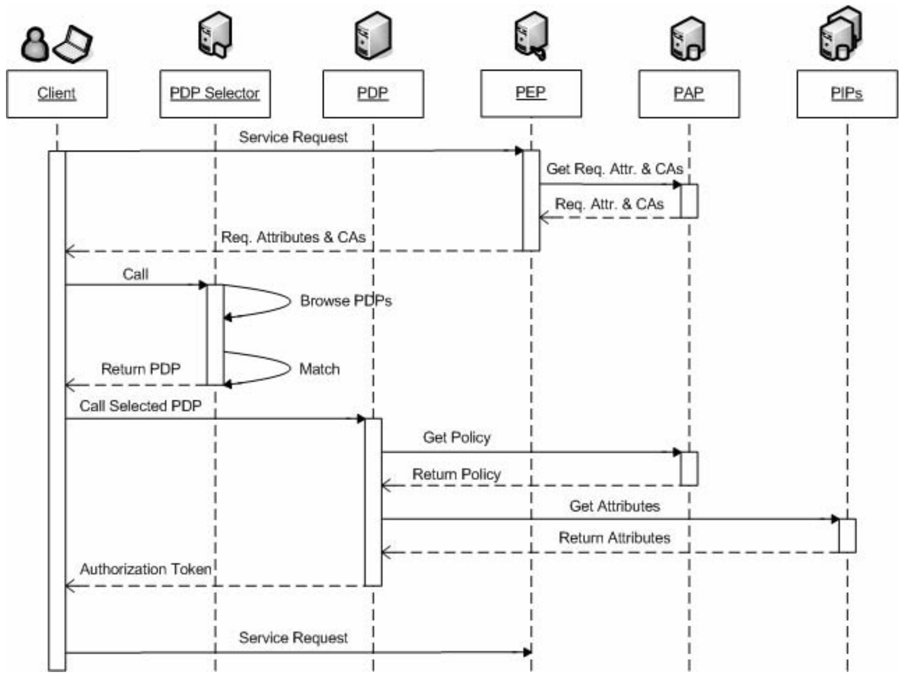


Fig. 2. Sequence diagram of the privacy-enhanced ABAC process

As the presented access control system targets the controlled disclosure of individual client attributes, the following will assume available privacy information at a low representation level.

Apart from privacy-related information on client side, the proposed infrastructure also relies on similar information pieces of the service provider, which needs to publish obligations regarding the client's data it commits itself to. This information represents the counterpart of privacy preferences of the client. As in our scenario a physically separated PDP is the target of client attributes and as the presented infrastructure facilitates multiple alternative PDPs, this information must be advertised by each available PDP.

A suitable candidate for the uniform encoding of privacy preferences of the client and obligations of the PDPs is XACML, which however is not built for the dynamic negotiation of policies, as the intersection of policies is not defined in the XACML standard [15]. The SOA-focused Web Service Profile of XACML (WS-XACML) directly addresses this shortcoming and offers a solution for the storing of the above mentioned information pieces. For a proper representation we utilize the two WS-XACML policy elements Requirements and Capabilities, which are wrapped into assertions, enabling a suitable matching in a web service environment.

Considering authorization information, the same WS-XACML elements can be used to map client attributes and access rules of the service provider. Table 1

Table 1. Requirements and capabilities of clients and PDPs

	Client	**PDPs (Service Provider)**
`ws-xacml:Requirements`	Privacy preferences, obligations to the service provider	Access Policy, rules built with client attributes
`ws-xacml:Capabilities`	Client attributes	Committed obligations of the service provider

categorizes privacy and authorization-related information into WS-XACML Requirements and Capabilities of the client and the PDPs, which in our case represent the service provider.

Requirements and Capabilities consist of XACML rules and policies expressed in a certain vocabulary. For our scenario they are wrapped in the following WS-XACML-specified privacy and authorization assertions, which are derived from the `XACMLAbstractAssertionType`:

– `XACMLPrivacyAssertion`
  The Requirements element of this assertion type issued by the client contains privacy preferences. As the Requirements element has to contain a Vocabulary element and as WS-XACML defines a P3P vocabulary, encoding the privacy preferences with the P3P vocabulary is a logical choice. The specification furthermore offers the client the possibility to define a Capabilities element, containing obligations the client is willing to accept. The privacy assertion of each PDP, on the other hand, expresses privacy-related obligations in the Capabilities field and might contain Requirements a client must meet.

– `XACMLAuthzAssertion`
  The authorization assertion's Requirements element filled by the PDP defines access rules and uses standard XACML attributes. According to the specification, every vocabulary is valid; XACML, however, already provides all needed elements. Like in the privacy assertion, the WS-XACML specification allows a Capabilities element to express the PDP's authorization-related obligations. Authorization assertions issued by the client contain personal attributes using the Capabilities field.

WS-XACML also describes an algorithm for matching assertion types. In our case, on the domain-specific level, matching is done between assertions of the same element name. The matching evaluates to "true", if all Requirements of one assertion can be fulfilled by Capabilities present in the other assertion. Certain special cases are also addressed; the interested reader is referred to [4] for detailed matching rules.

Once a suitable PDP has been selected using the privacy assertions of the client and those of all potential PDPs, the entitled PDP has to arrive at an access decision using the authorization assertions. If access to the requested service is granted, the PDP creates a `xacml-saml:XACMLAuthzDecisionStatement` token.

The token is passed to the service requestor which subsequently uses this token to gain access to the requested resource.

As mentioned earlier, an access control system that provides PDPs outside the service provider's premises must also incorporate trust-related security mechanisms. Choosing a malfunctioning PDP could result in the fraudulent or at least illegitimate access to a service. A malfunction in this context could either occur accidently in form of software bugs or deliberately through attacking the PDP. If, on the other hand, the user hands out his personal data to a malfunctioning PDP, his privacy might be violated or substantial financial losses have to be sustained, if an attacker manages to acquire some of the user's credentials.

One way to tackle this problem is to hardwire the system, effectively taking away every possibility of dynamic extension or reconfiguration. Therefore, we introduce a PKI scheme which, in practice, resembles the handling of the Transport Layer Security Protocol (TLS) [16]. In our scheme, the CAs attest that a particular PDP on a certain IP is allowed to evaluate certain attributes, by filling these information bits in the Distinguished Name field of the certificate and cryptographically signing it. Given that the service provider trusts that CA, implied by trusting the CAs root certificate, it can easily check if the digital certificate is valid and consequently the PDP is trustworthy. Such a PKI based scheme greatly increases the overall security of the dynamic PDP selections. Even the client can benefit from this PKI by using the same procedure to verify the validity of a certain PDP.

We point out that - following the concepts and technical aspects presented in this section - the whole access control process from request to response can be handled dynamically at runtime. No hard coding of the relationship between service provider and PDP is required, enabling the client to choose a privacy-conform PDP based on individual preferences.

4 Integration into Access-eGov

The European project Access-eGov[2] targets the interoperability of e-Government services by facilitating the composition of semantically annotated services to complex process definitions. Particular attention is paid to the fact that many e-Government services are not available online yet. Even those online services are rarely semantically annotated web services. For this reason, the whole platform is geared towards supporting offline and traditional e-services through the use of the specialized concept of a "complex goal" and an own process model.

A careful analysis of user requirements and a standard software development process led to a functional specification of the Access-eGov architecture's components [17,18]. Figure 3 gives an overview of the resulting service-oriented architecture. Users, service providers, and the Access-eGov platform are main actors, supported by management tools and specially crafted service ontologies. Users are represented by their digital personal assistant, which is responsible for invoking services and storing the state of execution. Apart from service-related

[2] http://www.access-egov.org/

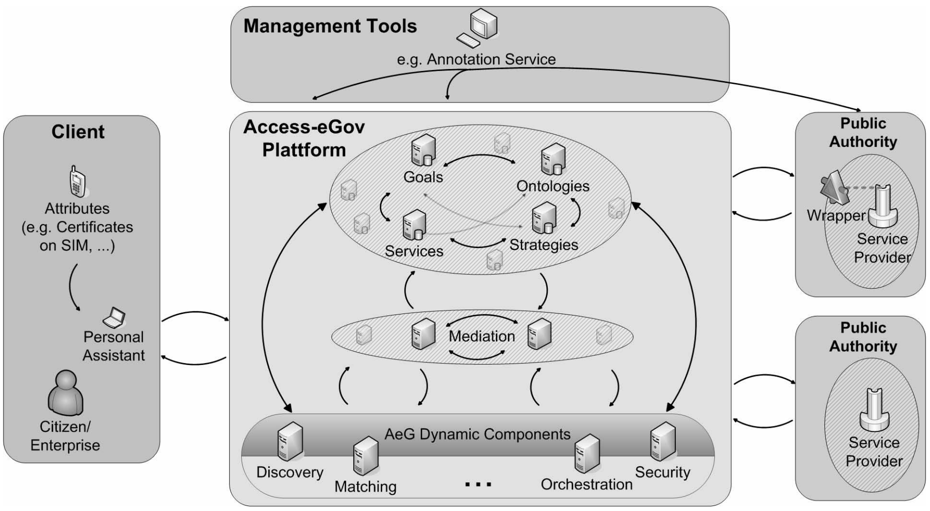


Fig. 3. Access-eGov service architecture

functions, the user's personal assistant maintains a user model containing profile information in the form of attributes. This profile also contains the user's privacy preferences and can be stored in a number of locations inside the Access-eGov platform or on user side, e.g. on a smartcard.

The dynamic components of Access-eGov operate on a peer-to-peer networked set of repositories, storing ontologies, annotated services, goals and strategies. A strategy represents the overall task a user wants to accomplish. The discovery and matching components split that task into well-defined sub-tasks, so called goals, which can be orchestrated and finally executed. The figure also shows the above mentioned option to integrate traditional e-services through the use of a wrapper. Once the discovery, matching and orchestration have been completed and a single service has been requested, the access control process introduced in Section 3 is started.

The personal assistant incorporates general as well as attribute-specific privacy preferences of the user in the corresponding XACML assertions. Utilizing the characteristics of a SOA-based peer-to-peer network, the security component in the platform does the same for the service provider's privacy obligations and requested attributes, which are part of a service repository. This enables a PDP Selector subcomponent to look up potential PDPs, either through brute force or through recursive use of the Access-eGov platform.

To single out a suitable PDP, the personal assistant tries to call the retrieved PDPs one by one. Once a PDP is found, which fulfills all the client's and provider's requirements, the provider checks its corresponding trust certificate. If the certificate is valid and issued by a CA the service provider trusts, the access request can be completed. In case of a successful completion, the personal assistant may get an Authorization Token, which then can be used by the client to access the originally requested service.

From an integration point of view, the best approach is to take the Authorization Token concept one step further. Making the government services and the Access-eGov platform a single-sign-on domain would greatly minimize the administrative effort and the need for redundant data storage. In the course of this project, it already showed, however, that single-sign-on solutions are not feasible with current local laws. The public authorities cannot abandon control of the authorization process for users of their services.

Targeting a flexible security infrastructure, the components of the Access-eGov security subsystem are modeled as services themselves. After semantically annotating them, using a specially crafted security ontology, those services are stored in the same service repositories as services offered by the public authorities. As the concept of a Goal in Access-eGov is not limited to serve users with specific tasks, goals can also be used to describe tasks and workflows unrelated to a specific user, but of a more technical nature. Above mentioned dynamic components can resolve a "security strategy" in exactly the same way, allowing the infrastructure to create new security systems on the fly.

5 Related Work

Focusing on ABAC and privacy aspects, our work builds on several research initiatives in the respective areas. In [19] the authors introduce a attribute certificate-based ABAC system using the AKENTI engine within the context of grid computing. Their approach, however, does not integrate any privacy-related mechanisms. On a theoretical level [20] describe a uniform framework that formulates and evaluates logical rules controlling service access and information release. Defining a powerful policy language, the uniform framework especially focuses on theoretical definitions of access control and information release policies and their logical matching. In [8] the authors enrich an ABAC system with an inference engine, targeting the semantic interoperability of security policies.

The definition of privacy preferences (disclosure rules of personal attributes) has been addressed by several PET initiatives [11,21]. Within the scope of the PRIME project the definition of data handling policies is proposed [22]. These policies define how personal data of users are dealt with at the receiving party. In that context, in [23] a privacy obligation management model is introduced, providing means to monitor and enforce privacy rules of end-users. In [24] XACML-based attribute release policies are utilized in the context of identity providers. Like our approach the author underscores the suitability of XACML to model attribute release policies. The author primarily addresses the controlled attribute release of an identity provider, not an access control infrastructure as a whole.

6 Conclusions

Distributed IT-infrastructures like the service-oriented architecture (SOA) increasingly rely on well fitted security mechanisms that protect both the privacy of clients and the resources of service providers. An attribute-based access control (ABAC) authorization system is flexible enough to satisfy the needs of a SOA. However, the ABAC model heavily relies on the disclosure of personal attributes, a characteristic that could conflict with privacy preferences of the client.

In this paper we introduced an ABAC system that provides a set of alternative, physically separated Policy Decision Points (PDPs). After the definition of individual privacy preferences (attribute disclosure rules), our approach facilitates a client to dynamically select a PDP that is in line with his privacy preferences. Addressing the necessary trust a physically separated PDP must provide, we embed a PKI in the access control process. For the process of defining, advertising and matching privacy preferences and PDP capabilities we take up concepts from the OASIS WS-XACML specification. Our approach enables the whole access control process to be handled dynamically at runtime. No static relations between a PDP and the service provider are necessary. The presented solution is part of a service-oriented security infrastructure within the research project Access-eGov.

Future work will involve performance and usability tests in the Access-eGov system. Furthermore, we are pursuing the notion of dynamically built PDP cascades, which will mitigate the effect of strict privacy and trust preferences of users and service providers, respectively. Finally, we are checking the potential of Trusted Computing technologies. With a trusted environment on client side the PDP could be moved directly to the client, an option that would obviously suit any privacy disclosure rule.

Acknowledgment

The work reported in this paper was in part funded by the European Union within the project Access-eGov (Sixth Framework Program, Contract No. FP6-2004-27020). We thank our project partners for helpful comments and stimulating discussions.

References

1. MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R.: Reference Model for Service Oriented Architecture 1.0. OASIS Standard (October 2006)
2. Yuan, E., Tong, J.: Attributed Based Access Control (ABAC) for Web Services. In: Proc. of the IEEE International Conference on Web Services (ICWS'05), Washington, DC, United States, pp. 561–569. IEEE Computer Society Press, Los Alamitos (2005)

3. Moses, T.: eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard (February 2005)
4. Anderson, A.: Web Services Profile of XACML (WS-XACML) Version 1.0. OASIS Working Draft, vol. 8 (December 2006)
5. Earp, J., Baumer, D.: Innovative Web Use to Learn About Consumer Behavior and Online Privacy. Communications of the ACM 46(4), 81–83 (2003)
6. Lopez, J., Oppliger, R., Pernul, G.: Authentication and Authorization Infrastructures (AAIs): A Comparative Survey. Computers & Security 23(7), 578–590 (2004)
7. Priebe, T., Dobmeier, W., Muschall, B., Pernul, G.: ABAC - Ein Referenzmodell für attributbasierte Zugriffskontrolle. In: Proc. of the 2nd Jahrestagung Fachbereich Sicherheit der Gesellschaft für Informatik (Sicherheit '05), Regensburg, Germany, pp. 285–296 (2005)
8. Priebe, T., Dobmeier, W., Kamprath, N.: Supporting Attribute-based Access Control with Ontologies. In: Proc. of the 1st International Conference on Availability, Reliability and Security (ARES '06), Washington, DC, United States, pp. 465–472. IEEE Computer Society Press, Los Alamitos (2006)
9. Nadalin, A., et al.: Web Services Security: SOAP Message Security 1.1. OASIS Standard Specification (2006)
10. World Wide Web Consortium: Web Services Policy 1.2 - Framework (WS-Policy). W3C Member Submission (April 2006)
11. Cranor, L., et al.: The Platform for Privacy Preferences 1.1 (P3P1.1) Specification. W3C Working Group Note (November 2006)
12. Cranor, L., Langheinrich, M., Marchiori, M.: A P3P Preference Exchange Language 1.0 (APPEL 1.0). World Wide Web Consortium Working Draft (April 2002)
13. Kolter, J., Schillinger, R., Pernul, G.: Building a Distributed Semantic-aware Security Architecture. In: Proc. of the 22nd International Information Security Conference (SEC2007), Sandton, South Africa, May 2007 (to Appear)
14. Anderson, A.: The Relationship Between XACML and P3P Privacy Policies (November 2004), http://research.sun.com/projects/xacml/XACML_P3P_Relationship.html
15. Andersson, A.: Sun Position Paper. W3C Workshop on Languages for Privacy Policy Negotiation and Semantics-Driven Enforcement (October 2006)
16. Dierks, T., Rescorla, E.: RFC 4346: The Transport Layer Security (TLS) Protocol Version 1.1. Internet RFCs (April 2006)
17. Klischewski, R., Ukena, S., Wozniak, D.: User Requirements Analysis & Development/Test Recommendation. Access-eGov deliverable D2.2 (July 2006)
18. Tomasek, M., Paralic, M., et al.: Access-eGov Components Functional Descriptions. Access-eGov deliverable D3.2 (November 2006)
19. Thompson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K., Essiari, A.: Certificate-based Access Control for Widely Distributed Resources. In: Proc. Proc. of the 8th USENIX Security Symposium, Washington, DC, United States (1999)
20. Bonatti, P., Samarati, P.: A Uniform Framework for Regulating Service Access and Information Release on the Web. Journal of Computer Security 10(3), 241–271 (2002)
21. Hansen, M., Krasemann, H.: Privacy and Identity Management for Europe PRIME White Paper. PRIME deliverable D15.1.d (July 2005)

22. Ardagna, C., De Capitani di Vimercati, S., Samarati, P.: Enhancing User Privacy Through Data Handling Policies. In: Proc. of the 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec 2006), Sophia Antipolis, France (July 2006)
23. Casassa Mont, M.: Towards Scalable Management of Privacy Obligations in Enterprises. In: Proc. of the Third International Conference on Trust, Privacy, and Security in Digital Business (TrustBus '06), Krakow, Poland, pp. 1–10(Septmeber 2006)
24. Hommel, W.: Using XACML for Privacy Control in SAML-Based Identity Federations. In: Communications and Multimedia Security, pp. 160–169 (2005)

A Scalable and Secure Cryptographic Service

Shouhuai Xu[1] and Ravi Sandhu[2]

[1] Department of Computer Science, University of Texas at San Antonio
shxu@cs.utsa.edu
[2] Institute for Cyber-Security Research, University of Texas at San Antonio
ravi.sandhu@utsa.edu

Abstract. In this paper we present the design of a scalable and secure cryptographic service that can be adopted to support large-scale networked systems, which may require strong authentication from a large population of users. Since the users may not be able to adequately protect their cryptographic credentials, our service leverages some better protected servers to help fulfill such authentication needs. Compared with previous proposals, our service has the following features: (1) it incorporates a 3-factor authentication mechanism, which facilitates compromise detection; (2) it supports immediate revocation of a cryptographic functionality in question; (3) the damage due to the compromise of a server is contained; (4) it is scalable and highly available.

Keywords: cryptographic service, scalability, security, compromise detection, compromise confinement, availability.

1 Introduction

Large-scale networked systems, such as peer-to-peer and grid systems, must be adequately protected; otherwise they may be abused or exploited to do more harm than good — Distributed Denial-of-Service (DDoS) attacks are just an example. An important aspect of secure large-scale networked systems is to enforce strong authentication, which would require a large population of users to utilize some cryptosystems such as digital signatures. Due to the very nature of cryptography, assurance offered by such authentications perhaps cannot be any better than security (or secrecy) of the corresponding cryptographic keys or functionalities. This is because compromise of a cryptographic key would allow the adversary to perfectly impersonate the victim user. The threat is amplified by the fact that average users often do not have the expertise or skill to secure their own computers, which may be justified by the fact that there have been many botnets that consist of many compromised computers.

Our contributions. We present the design of a scalable and secure cryptographic service that can be adopted to support large-scale networked systems, which require strong authentication from a large population of users. Specifically, our approach leverages some better protected servers to help protect the users' private signing keys and functionalities. Compared with a standard two-party threshold digital signature system (i.e., a user's private key is split into two shares such as one is stored on the user's machine and the other is stored on a remote server) and previous proposals for a similar purpose, our service has the following features:

* It incorporates a 3-factor authentication mechanism so that a signature may be produced in a certain way when a request: (1) presents a valid password (i.e., what you know); (2) is initiated from a party having access to the user's soft-token (i.e., what you have); (3) presents a valid fresh one-time secret (i.e., whether you *always* have access to the soft-token). As a result, the resulting service provides a *compromise detection* capability that may be of independent value.
* It supports a convenient *key disabling* that can be done using a standard username/password authentication. This is useful, for instance, when a user's device is stolen on a business trip because the user can disable its private key without having access to a backup of the content on its stolen device.
* The damage due to the compromise of a server is confined to a subset of the users subscribing to its service. Furthermore, the users associated with a compromised server do not have to re-initialize their private keys, unless they suspect that their own machines might have been compromised.
* It is scalable due to its "decentralized" nature (i.e., each server may serve a subset of users). It is highly available since a single server is enough to help a user fulfill its task. While we do not explore the details of utilizing the state machine approach [28] to securely replicate a server, it should not be difficult to extend our solution to fulfill such replications. In particular, in a related prior scheme we proposed [29], threshold cryptosystems have been adopted to fulfill distributed password-based authentication and signing.

Related prior work. The simplest approach to securing cryptographic keys is to let each user utilize some tamper-resistant hardware device. The industry has started to provide machines equipped with Trusted Platform Module (TPM) as specified by the Trusted Computing Group (`www.trustedcomputinggroup.com`). However, there are many legacy computers that need be better protected, meaning that alternate solutions are still useful.

One alternate approach to protecting cryptographic keys is to encrypt a key with a password; this is indeed widely deployed in real-life systems. However, the resulting security assurance is quite weak because, once a computer is compromised, the adversary can obtain the cryptographic keys without even conducting an off-line dictionary attack. Another approach is to let a user store its cryptographic key in a remote server, and download the key from the server after a password-based authentication [24]. This approach still allows the adversary, who can compromise a user's computer, to obtain the private key in question. Moveover, the remote server has to be trusted.

Our scheme follows the paradigm of "cryptography as a service" [13]. (This paradigm is different from the server-aided protocols [23,2], which were motivated to utilize a computationally powerful server to help a computationally poor device conduct expensive cryptographic computations.) In particular, we adopt as a starting point the proposal due to MacKenzie and Reiter [22], which follows [10,16]. The basic idea common to [22,10,16] as well as a similar result [9] is to split a private key into two shares such that one share (often called "soft-token") is stored on the user's device, and the other is stored on a remote server. These schemes have no single point of failure.

Finally, we should mention that there are some interesting cryptographic constructs that aim to protect private keys. Notable results include forward-security signatures [1,3,19], key-insulated signatures [15] and intrusion-resilient signatures [20]. The protections provided by these mechanisms are orthogonal to the protection provided by our approach. Nevertheless, they may be integrated together for a better protection.

Outline. In Section 2 we briefly review some necessary cryptographic preliminaries. In Section 3 we introduce the soft-token system model. In Section 4 we present a building block, which is utilized in Section 5 to construct the full-fledged service scheme. We conclude the paper in Section 6. Due to space limitation, the proofs of some theorems are deferred to the full version of the present paper [30].

2 Cryptographic Preliminaries

Let k be the primary security parameter (e.g., $k = 160$), and λ be a secondary security parameter for public keys (e.g., $\lambda = 1024$ means we use 1024-bit RSA moduli). A function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible if for any c there exists k_c such that $\forall k > k_c$ we have $\epsilon(k) < 1/k^c$. Let H (with an additional subscript as needed) be a hash function that, unless otherwise stated, is assumed to behave like a random oracle [5] with range $\{0,1\}^k$.

Pseudorandom functions. A pseudorandom function (PRF) family $\{f_v\}$ parameterized by a secret value v has the following property [17]: It is computationally infeasible to distinguish f_v, where v is uniformly chosen at random, from a random function (with the same domain and range).

Message authentication codes. We assume the standard property of message authentication codes (MACs): If the key a is unknown, then given multiple pairs $\langle m_i, MAC_a(m_i)\rangle$ where the m_i's may be adaptively chosen, it is computationally infeasible to compute any pair $\langle m, MAC_a(m)\rangle$ where $m \neq m_i$.

Public key cryptosystems. An public key cryptosystem $\mathcal{E}$ is a triple (GEN, E, D) of polynomial-time algorithms, where the first two are probabilistic. GEN, taking as input 1^λ, outputs a key pair (pk, sk). E, taking as input a public key pk and a message m, outputs an encryption c for m. D, taking as input a ciphertext c and a private key sk, returns a message m when c is valid and

$\perp$ otherwise. We assume an encryption scheme that is secure against adaptive chosen-ciphertext attack [26]. Basically, an attacker $\mathcal{A}$ is given pk and allowed to query the decryption oracle. At some point $\mathcal{A}$ generates two equal length strings X_0 and X_1 and sends them to a test oracle, which chooses $b \in_R \{0, 1\}$ and returns $Y = E_{pk}(X_b)$. Then $\mathcal{A}$ continues querying the decryption oracle, with the restriction that it cannot query the decryption of Y. Finally, $\mathcal{A}$ output b'. We say $\mathcal{A}$ succeeds if $b = b'$. Practical schemes are available [6,12].

Digital signature schemes. A digital signature scheme $\mathcal{S}$ is a triple (GEN, S, V) of polynomial-time algorithms, where the first two are probabilistic. GEN, taking as input 1^λ, outputs a key pair (pk, sk). S, taking as input a message m and a private key sk, outputs a signature σ for m. V, taking as input a message m, a public key pk, and a candidate signature σ, returns $b = 1$ if σ is a valid signature for m and $b = 0$ otherwise. We assume a signature scheme that is existentially unforgeable under adaptive chose-message attack [18]: a forger is given pk; it is allowed to query a signature oracle on messages of its choice; it succeeds if it outputs a valid signature for m that is not one of the messages signed before.

3 Model and Goals

System model. There are a set of users and a set of semi-trusted servers. As in a standard Public Key Infrastructure (PKI), a server has a pair of public and private keys, and so does a user. All the users and servers are probabilistic polynomial-time algorithms. A user splits its private key into two shares after a cryptographic transformation such that one share is stored on the user side (perhaps being encrypted with a password) and the other is stored on a remote server. A soft-token is a data structure a user stores. A soft-token (containing a user-side key share) may be stateful, so we may denote by $token^{(i)}$ the soft-token after the i^{th} transaction in which it is utilized.

A server has two interfaces: one for producing signatures and the other for disabling private keys. The resulting signatures can be verified using the public keys of the users, and thus can be used for authenticating the users in higher-layer applications. In order for a user to produce a signature, the user conducts an interaction with a server, which collaborates with the claimed user only when the user successfully authenticates itself to the server (not to the higher-layer application). In order for a user to disable its private key, the user needs to succeed in a certain authentication operation. A server maintains a database for recording relevant information that would allow the service provider to take actions (e.g., gathering payment when the service is payment-based).

Adversary. We consider an adversary who may have control over the network. The adversary may compromise certain resources including a user's soft-token, a user's password, and a server's private key. The adversary may break into a user's device when the client software is *active*; this explains why we consider an adversary that is strictly more powerful than the adversary considered in previous soft-token systems. We assume that the integrity of the server (e.g., the database) is guaranteed, even if an adversary can compromise the server's private key. This also models the situation where a semi-trusted server may be honest in performing the protocol, but curious about the users' private keys.

Goals. Recall that k is the primary security parameter. A cryptographic service should have the following properties:

* Abuse prevention. Consider a fixed pair of $\langle user, server\rangle$, where the $user$ possesses a soft-token $token$ and a password pwd. Denote by $\mathcal{ADV}(R)$ the type of adversary who succeeds in capturing the resource elements of $R \subseteq \{token, server, pwd\}$. When we say that an adversary $\mathcal{ADV}$ has access to $token$ we mean that $token$ is always available to $\mathcal{ADV}$; when we say that $\mathcal{ADV}$ has access to $\neg token^{(i)}$ we mean that $\mathcal{ADV}$ does not have access to $token^{(i)}$ but perhaps has access to $token^{(j)}$ for $0 \leq j < i$. Specifically,
  1. An adversary of type $\mathcal{ADV}\{server, pwd\}$ can only forge signatures that are valid with respect to the user's public key with a negligible probability in k.
  2. An adversary of type $\mathcal{ADV}\{token, server\}$ can forge signatures that are valid with respect to the user's public key only when the adversary succeeds in off-line dictionary guessing the user's password.
  3. An adversary of type $\mathcal{ADV}\{token\}$ can forge signatures that are valid with respect to the user's public key with probability negligibly more than $q/|\mathbb{D}|$ after q invocations of the server, where $\mathbb{D}$ is the dictionary from which the user's password is randomly drawn.
  4. An adversary of type $\mathcal{ADV}\{token, pwd\}$ can output — with only a probability negligible in k — signatures that are valid with respect to the user's public key after the user's private key is disabled.
  5. An adversary of type $\mathcal{ADV}\{\neg token^{(i)}, pwd\}$ can output — with only a probability negligible in k — signatures that are valid with respect to the user's public key after the user finishes the i^{th} transaction and before the user initiates the $(i+1)^{th}$ transaction.
* Compromise detection. The system itself can detect the compromise that an adversary has succeeded in impersonating the user for producing signatures.
* Immediate revocation. A user can request a server to disable its private key by executing a standard username/password authentication.
* Compromise confinement. The impact due to the compromise of a server is contained to a *subset* of the users subscribing to its service. Moreover, these users do not have to re-initialize their private keys.
* Scalability. The system can serve a large population of users.
* High availability. The system is highly available, even if some servers are under DDoS attacks.

4 Building Block: A Single Server Soft-Token Scheme

In this section we present a building block, which is extended from a scheme presented in [22] and will be incorporated into our full-fledged scheme in Section 5.

Suppose the server's public data (e.g., public key) are available to the users, and consider a fixed pair of $\langle user, server\rangle$. The user runs the initialization process to generate a soft-token $token^{(0)}$ using its public and secret data as well as the server's public data. In the i^{th} transaction ($i = 1, 2, \cdots$), the user, who has access to $token^{(i-1)}$ that is generated in the $(i-1)^{th}$ transaction or in the initialization process when $i = 1$, signs a message by interacting with the server. The server collaborates with the claimed user only when the user presents the password and the state information ϑ chosen by the server in the last transaction. At the end of the i^{th} transaction, the user obtains the signature, generates a new token $token^{(i)}$ using the cryptographic state information ϑ chosen by the server, and erases $token^{(i-1)}$. The server tracks the changes of the ϑ's in its database.

Denote a user's public key by $pk_{user} = \langle e, N\rangle$ and private key by $sk_{user} = \langle d, N, \phi(N)\rangle$, where $ed = 1 \bmod \phi(N)$, N is the product of two large prime numbers, and ϕ is the Euler totient function. In the standard encode-then-sign paradigm, the signature sig on message m is $S_{\langle d,N,\phi(N)\rangle}(m) = \langle r, s\rangle$, where $r \in_R \{0,1\}^{len_{pad}}$, $s = (\mathsf{encode}(m, r))^d \bmod N$ for some encoding function encode. A signature $\langle r, s\rangle$ can be verified by checking if $s^e = \mathsf{encode}(m, r) \bmod N$. The function encode could be either deterministic (e.g., $len_{pad} = 0$ in the case of hash-and-sign [14]) or probabilistic (e.g., PSS [7]), these types of signatures were proven secure against adaptive chosen-message attacks in the random oracle model. The basic idea for splitting the private key d is to let $d_1 + d_2 = d \bmod \phi(N)$. The scheme has the following components.

Token initialization. Suppose $H_1 : \{0,1\}^* \longrightarrow \{0,1\}^k$ and $f : \{0,1\}^* \rightarrow \{0,1\}^{\lambda+k}$. The inputs are the server's public encryption key pk_{server}, the user's password pwd, the user's public key $pk_{user} = \langle e, N\rangle$ and private key $sk_{user} = \langle d, N, \phi(N)\rangle$. The initialization proceeds as follows.

$$\begin{array}{lll}
uid & = & username \\
v & \in_R & \{0,1\}^k \\
a & \in_R & \{0,1\}^k \\
b & = & H_1(pwd) \\
d_1 & = & f(v, pwd) \\
d_2 & = & d - d_1 \bmod \phi(N) \\
\tau & = & E_{pk_{server}}(\langle a, b, uid, d_2, N\rangle) \\
ct & = & 0 \\
st & \in_R & \{0,1\}^k \\
token & = & (ct, st, v, a, \tau, e, N, pk_{server})
\end{array}$$

The user chooses its own uid, a unique and memorizable string such as its email address. The soft-token is $token = (ct, st, v, a, \tau, e, N, pk_{server})$, where ct is an incremental counter indicating the serial number of a transaction (which is used for simplifying the description), st is the state information that will be chosen by the server (for the time being of $ct = 0$, it is just a placeholder). All the other values, including b, d, d_1, d_2, $\phi(N)$, and pwd, are erased.

Server database. The server maintains a database of $\Upsilon = (\tau, uid, count, \vartheta, m, r)$, where uid is obtained after receiving the first service request, and $count$ is an incremental counter (with initialized value zero). Each record of the database represents a transaction corresponding to $\tau = E_{pk_{server}}(\langle a, b, uid, d_2, N\rangle)$. The database has two operations: $\mathsf{append}(\tau, uid, count, \vartheta, m, r)$ for appending a record $(\tau, uid, count, \vartheta, m, r)$ to the database, and $\mathsf{last}(\tau, uid, count, \vartheta, m, r)$ for returning either the record $(\tau, uid, count, \vartheta, m, r)$ corresponding to the *last* transaction corresponding to τ or NULL (meaning that the token corresponding to τ has never been used before).

Signing protocol. The client software prompts the user to enter the password pwd, to get the to-be-signed message m as well as the soft-token $token = (ct, st, v, a, \tau, e, N, pk_{server})$. The protocol is depicted in Fig. 1.

USER		SERVER
$token = (ct, st, v, a, \tau, e, N, pk_{server})$		
$\beta = H_1(pwd)$, $\rho_1 \in_R \{0,1\}^k$		
$\rho_2 \in_R \{0,1\}^\lambda$, $\rho_3 \in_R \{0,1\}^k$		
$r \in_R \{0,1\}^{len_{pad}}$		
$\gamma = E_{pk_{server}}(\langle m, r, \beta, st, \rho_1, \rho_2, \rho_3\rangle)$		
$\delta = MAC_a(\langle \gamma, \tau\rangle)$		
	$\xrightarrow{(\gamma, \tau, \delta)}$	
		abort IF τ has been disabled
		$\langle a, b, uid, d_2, N\rangle = D_{sk_{server}}(\tau)$
		abort IF $MAC_a(\langle \gamma, \tau\rangle) \neq \delta$
		$\Upsilon = \mathsf{last}(\tau, uid, count, \vartheta', m', r')$
		$\langle m, r, \beta, st, \rho_1, \rho_2, \rho_3\rangle = D_{sk_{server}}(\gamma)$
		abort IF $\beta \neq b$
		abort IF $\Upsilon \neq \mathsf{NULL} \wedge st \neq \vartheta'$
		$\sigma = (\mathsf{encode}(m, r))^{d_2} \bmod N$
		$\vartheta \in_R \{0,1\}^k$, $\theta = \vartheta \oplus \rho_1$
		$\eta = \sigma \oplus \rho_2$, $\varphi = MAC_{\rho_3}(\langle \theta, \eta\rangle)$
		$count = count + 1$
		$\mathsf{append}(\tau, uid, count, \vartheta, m, r)$
	$\xleftarrow{(\theta, \eta, \varphi)}$	
abort IF $MAC_{\rho_3}(\langle \theta, \eta\rangle) \neq \varphi$		
and $\sigma = \eta \oplus \rho_2$ and $d_1 = f(v, pwd)$		
$s = \sigma \cdot (\mathsf{encode}(m, r))^{d_1} \bmod N$		
abort IF $s^e \neq \mathsf{encode}(m, r)$		
$st = \theta \oplus \rho_1$, $ct = ct + 1$		
$token' = (ct, st, v, a, \tau, e, N, pk_{server})$		
erase β, d_1, ρ_1, ρ_2, ρ_3, θ, η, φ, ϑ, $token$		

Fig. 1. Building block: a single-server scheme with stateful soft-tokens

Let us briefly explain the functions of the protocol elements:

* β is a value showing that the user knows the password *pwd*.
* ρ_1 and ρ_2 are two one-time pads chosen by the user, and will be used by the server to encrypt the state information ϑ and the partial signature σ (produced using the partial private key d_2), respectively.
* ρ_3 is a one-time message authentication key that allows the user to detect compromise of *token* because the adversary could tamper with θ while keeping η intact.
* r is a len_{pad}-bit random string used in the encoding function.
* γ is the encryption of m, r, β, st, ρ_1, ρ_2, and ρ_3.
* δ and φ are message authentication codes computed using a and ρ_3, respectively. (Note that both δ and φ are not for preventing abuses, but for detecting attacks.)

Key disabling protocol. In order to disable its private key, the user authenticates itself to the server by conducting a standard username/password authentication protocol corresponding to *uid*/*pwd*. The server will query its database to get $b = H_1(pwd)$ from $\langle a, b, uid, d_2, N\rangle = D_{sk_{server}}(\tau)$, where τ corresponds to *uid*. Any secure password protocol (e.g., [8,4,11,21]) can be used for this purpose.

4.1 Discussions

On 3-factor authentication. Previous key-split schemes (such as [22]) employ a 2-factor authentication mechanism based on a password and a soft-token. Whereas, we employ a 3-factor authentication mechanism so that a signature is produced when a request (i) presents a valid password (i.e., what you know), (ii) is initiated from a party having access to the user's soft-token (i.e., what you have), and (iii) presents a valid fresh one-time secret (i.e., whether you *always* have access to the soft-token). The last factor helps achieve the newly introduced compromise detection and the enhanced abuse prevention (see Section 4.2).

On atomicity of the transactions. We assumed that atomicity of the transactions is ensured. This may be problematic when, for example, the servers are under a DDoS attack. This issue is addressed via another layer of assurance for synchronization in Section 5.

On light-weight key disabling. Allowing a user to disable its private key via a standard username/password authentication has the advantage that a user does not have to resort to its soft-token, which may not be available (e.g., when the user's device is stolen). Although this convenience seemingly gives an adversary the chance to impose denial-of-service attack (i.e., the adversary can request the server to disable the user's private key), we argue that there are no sever consequences. First, suppose an adversary does not know a user's token or password. Then, the adversary can conduct an on-line dictionary attack against the user's password. This is sever if the on-line dictionary attack can be launched automatically by a software program. Fortunately, there exist some effective methods (e.g., [25]) to force the adversary to conduct a *manual* on-line dictionary attack, which may be unlikely. Second, suppose an adversary knows a user's token but not password. Then, it is seemingly more attractive for the adversary to manage to get the user's password so that it can produce signatures (rather than disable the user's private key). Moreover, the user might also have to disable its private key once the user realized that its soft-token has been compromised. Third, suppose an adversary knows a user's password but not token. Then, the adversary can always disable the user's private key. This may not be seen as a drawback because the user has to disable its private key once the user realized that its password has been compromised. Fourth, suppose an adversary knows a user's token and password. Then, the adversary is already able to produce signatures by contacting the server, which is perhaps more attractive than to conduct a denial-of-service attack by disabling the victim user's private key.

4.2 Analysis

Our scheme does not incur any significant extra complexity, when compared with the starting-point scheme in [22]. Specifically, a soft-token keeps some state information (e.g., 160 bits), and a server keeps some state information linear to the number of users (which can be easily mitigated by letting the server use a pseudorandom function). Moreover, no extra exponentiations are imposed on a user or a server. Below we analyze the security properties.

Proposition 1. *The single server scheme implements* some *of the requirements specified in Section 3 (the others will be fulfilled in the full-fledged scheme via another layer of protection).*

* Abuse prevention*. This is analyzed in Theorems 1-5.*
* Compromise detection*. Suppose atomicity of the transactions and integrity of the server are guaranteed. Lack of synchronization means that either an adversary had succeeded in impersonating the user, or the token had been tampered with.*
* Immediate revocation*. This is true since the request for disabling a private key is authenticated by pwd, whereas uid is also remembered by the user.*

In order to prove the abuse prevention, we introduce the following formal security model. Denote D-RSA$[\mathcal{E}, \mathbb{D}]$ the real-world single-server signing system based on an encryption scheme $\mathcal{E}$ for the server and dictionary $\mathbb{D}$. An adversary is given $\langle e, N\rangle$ where $(\langle e, N\rangle, \langle d, N, \phi(N)\rangle) \longleftarrow GEN_{RSA}(1^\lambda)$, the public data generated in the initialization procedure, and certain secret data of the user and/or server (depending on the type of the adversary). The goal of the adversary is to forge RSA signatures with respect to $\langle e, N\rangle$. The adversary is allowed to have the following types of oracle queries:

1. start(m) – This results in a user to initiate the protocol. The oracle may execute according to the protocol, maintain state as appropriate (i.e., there is an implicit notion of sessions), and return (γ, τ, δ).

2. $\mathsf{serve}(\gamma, \tau, \delta)$ – This represents the receipt of a message ostensively from the user. The oracle may execute according to the protocol to return (θ, η, φ).
3. $\mathsf{finish}(\theta, \eta, \varphi)$ – This represents the receipt of a response ostensively from the server. The oracle may execute according to the protocol to return a valid signature.

An adversary of type $\mathcal{ADV}\{server, pwd\}$, $\mathcal{ADV}\{token, server\}$, or $\mathcal{ADV}\{token\}$ succeeds in breaking the scheme if it can output a valid signature $\langle r, s\rangle$ on message m and there was no $\mathsf{start}(m)$ query. An adversary of type $\mathcal{ADV}\{token, pwd\}$ succeeds in breaking the scheme if it can output a valid signature $\langle r, s\rangle$ on message m and there was no $\mathsf{serve}(\gamma, \tau, \delta)$ query, where $\langle m, *, *, *, *, *, *\rangle = D_{sk_{server}}(\gamma)$. An adversary of type $\mathcal{ADV}\{\neg token^{(i)}, pwd\}$ succeeds in breaking the scheme if it can output a valid signature $\langle r, s\rangle$ on message m after the user finishes the i^{th} transaction and before the user initiates the $(i+1)^{th}$ transaction, and there was no $\mathsf{serve}(\gamma, \tau, \delta)$ query such that $\langle m, *, *, *, *, *, *\rangle = D_{sk_{server}}(\gamma)$.

Denote by q_{user} the number of $\mathsf{start}(\cdot)$ queries to the user, q_{server} the number of $\mathsf{serve}(\cdot,\cdot,\cdot)$ queries to the server. Let q_h and q_f be the number of queries to the random oracles h and f, respectively. Let q_o be the number of other oracle queries not counted above. Let $\bar{q} = (q_{user}, q_{server}, q_h, q_f, q_o)$. We also denote by $|\bar{q}| = q_{user} + q_{server} + q_h + q_f + q_o$ as the total number of oracle queries.

We say an adversary $(\bar{q}, \varepsilon)$-breaks D-RSA if it makes $\bar{q}$ oracle queries (of the respective types and to the respective oracles) and succeeds with probability at least ε. In the following, "$\approx$" means equality within negligible factors.

We say an attacker $\mathcal{A}$ (q, ε)-breaks $\mathcal{E}$ if the attacker makes q queries to the decryption oracle and $2 \cdot \Pr[\mathcal{A}$ succeeds$] - 1 \geq \varepsilon$, which implies $\Pr[\mathcal{A}$ outputs $0|b = 0] - \Pr[\mathcal{A}$ outputs $0|b = 1] \geq \varepsilon$. Note that if $\mathcal{E}$ uses random oracles, the oracles may be queried by the attacker along with the encryption oracle.

We say a forger (q, ε)-breaks a signature scheme if it makes q queries and succeeds with probability at least ε. Note that if $\mathcal{S}$ uses random oracles, the oracles may be queried by the forger along with the signature oracle.

Now we present the theorems for the $\mathsf{abuse\ prevention}$ property of the single-server signing protocol. Theorem 1-4 are extended from [22] and their proofs are deferred to the full version of the present paper [30] (due to space limitation).

Theorem 1. *Suppose $\{f_v\}$ is a pseudorandom function family. If an adversary $\mathcal{F}$ of type $\mathcal{ADV}\{server, pwd\}$ can $(\bar{q}, \varepsilon)$-break D-RSA$[\mathcal{E}, \mathbb{D}]$ system, then there exists a forger $\mathcal{F}^*$ able to (q_{user}, ε')-break the underlying RSA signature scheme, where $\varepsilon' \approx \varepsilon$.*

Theorem 2. *Let H_1 and f be random oracles. If $\mathcal{F}$ of type $\mathcal{ADV}\{token, server\}$ can $(\bar{q}, \varepsilon)$-break D-RSA$[\mathcal{E}, \mathbb{D}]$ system, there exists a forger $\mathcal{F}^*$ able to (q_{user}, ε')-break the underlying RSA signature scheme, where $\varepsilon' \approx \varepsilon - \frac{q_h + q_f}{|\mathbb{D}|}$.*

Theorem 3. *Suppose H_1 has a negligible probability of collision over $\mathbb{D}$. If an adversary $\mathcal{A}$ of type $\mathcal{ADV}\{token\}$ can $(\bar{q}, \varepsilon)$-break the D-RSA$[\mathcal{E}, \mathbb{D}]$ system for $\varepsilon = \frac{q_{server}}{|\mathbb{D}|} + \psi$, then there exists either a forger $\mathcal{F}^*$ able to (q_{user}, ε')-break* *the underlying RSA signature scheme for $\varepsilon' \approx \frac{\psi}{2}$, or an attacker $\mathcal{A}^*$ able to $(2q_{server}, \varepsilon'')$-break $\mathcal{E}$ for $\varepsilon'' \approx \frac{\psi}{2(1+q_{user})}$.*

Theorem 4. *Suppose the underlying RSA signature scheme is deterministic. If an adversary $\mathcal{A}$ of type $\mathcal{ADV}\{token, pwd\}$ can $(\bar{q}, \varepsilon)$-break the D-RSA$[\mathcal{E}, \mathbb{D}]$ system, then there exists either a forger $\mathcal{F}^*$ able to $(q_{server}, \varepsilon')$-break the underlying RSA signature scheme for $\varepsilon' \approx \frac{\varepsilon}{2}$, or an attacker $\mathcal{A}^*$ able to $(2q_{server}, \varepsilon'')$ -break $\mathcal{E}$ for $\varepsilon'' \approx \frac{\varepsilon}{2(1+q_{user})}$.*

Theorem 5. *Suppose the underlying RSA signature scheme is deterministic. If an adversary $\mathcal{A}$ of type $\mathcal{ADV}\{\neg token^{(i)}, pwd\}$ can $(\bar{q}, \varepsilon)$-break the D-RSA$[\mathcal{E}, \mathbb{D}]$ system, then there exists either a forger $\mathcal{F}^*$ able to $(q_{server}, \varepsilon')$-break the underlying RSA signature scheme for $\varepsilon' \approx \frac{\varepsilon}{2q_{user}}$, or an attacker $\mathcal{A}^*$ able to $(2q_{server}, \varepsilon'')$-break $\mathcal{E}$ for $\varepsilon'' \approx \frac{\varepsilon}{4q_{user}}$.*

Proof. (sketch) Suppose there exists i, $1 \leq i \leq q_{user}$, such that $\mathcal{A}$ outputs a valid signature for a new message with probability at least ε after the user finishes the i^{th} transaction and before the user initiates the $(i+1)^{th}$ transaction. Consider an algorithm $\mathcal{F}^{**}$ that is the same as $\mathcal{F}^*$ in Theorem 4, except that:

* Let τ be the encryption of 0-string of appropriate length, the first $i-1$ γ's be the encryptions of normal messages, and the i^{th} γ be the encryption of 0-string of appropriate length.
* A token is sent to $\mathcal{A}$ only when it issues a get() query, which is not allowed for the i^{th} token generated by the user in the i^{th} transaction. Also, $\mathcal{A}$ can maintain the consistency between the user side and the server side by issuing a coordinate(*token*) query.

Given the simulation generated by $\mathcal{F}^{**}$, if Pr[$\mathcal{A}$ succeeds] $\geq \frac{\varepsilon}{2}$, then it is clear that there exists $\mathcal{F}^*$ able to $(q_{server}, \varepsilon')$-break the underlying RSA signature scheme, where $\varepsilon' \approx \frac{\varepsilon}{2q_{user}}$; otherwise, there exists $\mathcal{A}^*$ able to $(2q_{server}, \varepsilon'')$-break $\mathcal{E}$, where $\varepsilon'' \approx \frac{\varepsilon}{4q_{user}}$.

Consider an algorithm $\mathcal{A}^{**}$ that is given a public key pk'. Now, $\mathcal{A}^{**}$ can *perfectly* simulate the real-world system as follows. It generates the pair of public and private keys on behalf of the user. It need have access to the decryption oracle, but at most $2q_{server}$ times. Note that $\mathcal{A}$ is given a newly generated token only when it issues a get() query, and that no get() query is allowed after the user finishes the i^{th} transaction and before the user initiates the $(i+1)^{th}$ transaction. Therefore, $\mathcal{A}$ succeeds in forging with probability at least ε.

Now consider the same simulation, except that τ is the encryption of 0-string of appropriate length, the first $i-1$ γ's are the encryptions of normal messages, and the i^{th} γ is the encryption of 0-string of appropriate length. This simulation is equivalent to the simulation of $\mathcal{F}^{**}$. Therefore, Pr[$\mathcal{A}$ succeeds] $< \frac{\varepsilon}{2}$.

A standard hybrid argument shows that $\mathcal{A}^{**}$ can $(2q_{server}, \frac{\varepsilon}{4})$-break $\mathcal{E}$, which means that there exists $\mathcal{A}^*$ able to $(2q_{server}, \frac{\varepsilon}{4q_{user}})$-break $\mathcal{E}$. □

5 Full-Fledged Scheme

Now we present the full-fledged scheme, which is built on top of the building-block discussed in the previous section. The basic idea underlying the scheme is the following. In order to achieve compromise confinement, we augment $b = H_1(pwd)$ to $b = H_1(c, pwd)$, where c is a cryptographic secret used to ensure that $b = H_1(c, pwd)$ itself does not leak any significant information of a password pwd. A drawback of this approach is that a user cannot disable its private key using a standard username/password authentication mechanism. Nevertheless, this can be resolved by letting a user choose two passwords, one for generating signatures and the other for disabling its key. Specifically, this can be done by further augmenting $\tau = E_{pk_{server}}(\langle a, b, uid, d_2, N\rangle)$ to $\tau = E_{pk_{server}}(\langle a, b, b^*, uid, d_2, N\rangle)$ such that $b = H_1(c, pwd)$ and $b^* = H_2(\pi)$, where password pwd is for generating signatures, and password π is for disabling the key.

Special care is also taken to address atomicity and availability issues. Suppose a server $server_i$ cannot finish a transaction within a certain time interval. Then, it is reasonable to allow the user to contact another server. This flexibility complicates the facilitation of transaction atomicity. We resolve this issue by incorporating a simple "commit/rollback" mechanism. Moreover, a *commit* or *rollback* request should be authenticated. This can be done by augmenting $\gamma = E_{pk_{server}}(\langle m, r, \beta, st, \rho_1, \rho_2, \rho_3\rangle)$ to $\gamma = E_{pk_{server}}(\langle m, r, \beta, st, \rho_1, \rho_2, \rho_3, \rho_4, \rho_5\rangle)$, where $\rho_4 = H_3(\rho_4^*)$ is used for committing a transaction and $\rho_5 = H_3(\rho_5^*)$ is used for rollbacking a transaction. The scheme has the following components.

Token initialization. Suppose $H_1 : \{0,1\}^* \longrightarrow \{0,1\}^k$, $H_2, H_3 : \{0,1\}^k \longrightarrow \{0,1\}^k$, and $f : \{0,1\}^* \rightarrow \{0,1\}^{\lambda+k}$ are appropriate hash and pseudorandom functions, respectively. A user chooses two passwords – pwd for generating signatures and π for disabling the private key, and a pair of public and private keys $(pk_{user} = \langle e, N\rangle; sk_{user} = \langle d, N, \phi(N)\rangle)$. A user chooses n servers as its service providers, and each server has a public key $pk_{server,i}$, where $1 \leq i \leq n$. The initialization process proceeds as follows.

$$\begin{array}{lll}
uid & = & username \\
v_i & \in_R & \{0,1\}^k,\ 1 \leq i \leq n \\
a_i & \in_R & \{0,1\}^k,\ 1 \leq i \leq n \\
c_i & \in_R & \{0,1\}^k,\ 1 \leq i \leq n \\
b_i & = & H_1(c_i, pwd),\ 1 \leq i \leq n \\
b^* & = & H_2(\pi) \\
d_{1,i} & = & f(v_i, pwd),\ 1 \leq i \leq n \\
d_{2,i} & = & d - d_{1,i} \bmod \phi(N),\ 1 \leq i \leq n \\
\tau_i & = & E_{pk_{server,i}}(\langle a_i, b_i, b^*, uid, d_{2,i}, N\rangle),\ 1 \leq i \leq n \\
ct_i & = & 0,\ 1 \leq i \leq n \\
st_i & \in_R & \{0,1\}^k,\ 1 \leq i \leq n \\
token_i & = & (ct_i, st_i, v_i, a_i, c_i, \tau_i, e, N, pk_{server,i}),\ 1 \leq i \leq n
\end{array}$$

The user keeps n soft-tokens on its device and erases all the other values.

Server databases. Each server, $server_i$ for $1 \leq i \leq n$, maintains a database of $\Upsilon_i = (\tau, uid, count, \vartheta, m, r)$, where uid is obtained after receiving the first service request, *count* is an incremental counter (with initialized value zero) maintained by the server. Each record of the database represents a transaction with respect to $\tau = E_{pk_{server}}(\langle a, b, b^*, uid, d_2, N\rangle)$, There are two types of operations regarding the database. First, $\mathsf{append}(\tau, uid, count, \vartheta, m, r)$ appends $(\tau, uid, count, \vartheta, m, r)$ into the database. Second, $\mathsf{last}(\tau, uid, count, \vartheta, m, r)$ returns either $(\tau, uid, count, \vartheta, m, r)$ corresponding to the *last* transaction and τ, or NULL (meaning that the token corresponding to τ has never been used before).

Signing protocol. The protocol is depicted in Figure 2. The user first sends a request to a (randomly chosen) server. If the transaction can not be finished within certain time, the user contacts another server. Note that such a switching process can be made transparent to the user.

```
USER                                                        SERVER_1, ..., SERVER_n
$token_j = (ct_j, st_j, v_j, a_j, \tau_j, e, N, pk_{server,j})$, $1 \le j \le n$
$trial = \phi$
REPEAT if a transaction cannot be finished within time $\Delta$
  choose $i \in \{1, 2, \cdots, n\} - trial$ according to a policy (e.g., random)
  $trial = trial \cup \{i\}$
  user provides $token_i$ and $\gamma = E_{pk_{server,i}}(\langle m, r_i, \beta_i, st_i, \rho_{1,i}, \rho_{2,i}, \rho_{3,i}, \rho_{4,i}, \rho_{5,i}\rangle)$
    The server executes as follows:
      IF $st_i$ matches the last "committed" transaction, run the single server protocol
      IF $st_i$ matches the last "pending" transaction,
          change its status as "committed" and execute as specified;
        ELSE reject the request and inform "user compromised."
UNTIL there is a successful transaction using $token_j$
$\alpha = MAC_{\rho_{3,j}}(\text{"commit"}, \tau_j, \rho^*_{4,j})$
$token_j$ is updated
                $\xrightarrow{(\text{"commit"}, \tau_j, \rho^*_{4,j}, \alpha)}$
                                          $j^{th}$ server commits the transaction
$trial = trial - \{j\}$
FOR each $i \in trial$
  $\alpha = MAC_{\rho_{3,i}}(\text{"rollback"}, \tau_i, \rho^*_{5,i})$
                $\xrightarrow{(\text{"rollback"}, \tau_i, \rho^*_{5,i}, \alpha)}$
                                          $i^{th}$ server rollbacks the transaction
```

Fig. 2. The full-fledged scheme

Key disabling protocol. In order to disable its private key, the user authenticates itself to each of the n servers by proving that it knows the password π corresponding to *uid*. This process is the same as in the underlying single server protocol, and can be made automatic via an appropriate software design (for ease of deployment).

5.1 Analysis and Discussion

Since transaction atomicity plays an important role in our system, we must show that it has no significant security consequence if atomicity is violated.

Proposition 2. *Suppose there is no system crash resulting in the loss of system state information. Suppose a server keeps a database of state information* $(\tau, uid, count, \vartheta, m, r)$. *Then there is no denial-of-service attack because of an out-of-synchronization between a user and a server.*

Proof. Recall that a server classifies transactions into two categories: "committed" and "pending". Note that a successfully rollbacked transaction is treated as a "committed" one, but corresponding to the last successfully committed transaction. This is so because that a rollbacked transaction can be viewed as one where no actions have been taken whatsoever. So, when a user sends a request with certain state information ϑ, there are three cases.

* ϑ matches the state information corresponding to a "committed" transaction $\mathcal{T}$. There are further two cases.
  1. $\mathcal{T}$ is the last "committed" transaction. The This is the normal case and the protocol proceeds as specified.
  2. $\mathcal{T}$ is not the last "committed" transaction. The server simply rejects this request. In this case, the server could inform the user, perhaps via an out-of-band channel, that the user side might have been compromised. This is so because in our design a user updates its state information before sending a commitment request.

* ϑ matches the state information corresponding to a "pending" transaction $\mathcal{T}$. There are further two cases.
  1. $\mathcal{T}$ is the last "pending" transaction. This means that the server did not commit the transaction yet. Then the server can simply accept the request and change the "pending" transaction into a "committed" one.
  2. $\mathcal{T}$ is not the last "committed" transaction. This is impossible, because our design ensures that whenever a server accepts a request (and sends new state information to a user), it always treat the *last* transaction as committed.

* ϑ matches no state information in the server database at all. The server simply rejects the request. In this case, the server could inform the user, perhaps via an out-of-band channel, that the user side might have been compromised. This is so because in our design we let a server choose the state information.

□

The above proposition implies that we can treat all transactions as atomic. Now we are ready to look at the properties of the full-fledged scheme. We claim that the full-fledged scheme implements all of the goals specified in Section 3. Informally, we observe the following:

* Abuse prevention. This can be formally analyzed by extending the theorems in the underlying single server scheme.

* Compromise detection. Suppose atomicity of the transactions and integrity of the server are guaranteed. Lack of synchronization means either an adversary had successfully impersonated the user, or the token had been tampered with. In either case, the user needs to disable its private key.
* Immediate revocation. This is true since the request for disabling a private key is authenticated by π, whereas *uid* is also remembered by the user.
* Compromise confinement. Once it is known that a server has been compromised, only the users who suspects that their tokens have been compromised need to re-initialize their keys, whereas the others just need to erase their tokens corresponding to the compromised server (i.e., no need to re-initialize their private keys). We notice that when a user knows that its password for disabling its private key, namely π, may have been compromised (e.g., due to the compromise of a server), the user can, if desired, execute an appropriate password-change protocol (e.g., the one associated with the adopted password authentication scheme) to update π to some π'. Such a process would be much more light-weight than a process for updating private keys. Note, however, that the password update process is not necessary from the perspective of the security of the user's signature scheme.
* Scalability. The system is scalable because the servers are decentralized, namely that the servers are operated by possibly many service providers.
* High availability. The system is highly available since a single server is sufficient for the users to generate signatures.

6 Conclusion and Future Work

We presented a scalable and secure cryptographic service, which has the following features: (1) it incorporates a 3-factor authentication mechanism; (2) it supports immediate revocation of a cryptographic key or functionality in question; (3) the damage due to the compromise of a server is contained; (4) it is scalable and highly available.

Acknowledgement. We thank Philip MacKenzie for valuable feedbacks, and the anonymous reviewers for their comments.

This work was supported in part by NSF and UTSA.

References

1. Anderson, R.: Invited Talk at ACM CCS'97.
2. Asokan, N., Tsudik, G., Waidner, M.: Server-Supported Signatures. Journal of Computer Security 5(1) (1997)
3. Bellare, M., Miner, S.: A forward-secure digital signature scheme. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, Springer, Heidelberg (1999)
4. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, Springer, Heidelberg (2000)

5. Bellare, M., Rogaway, P.: Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols. In: Proc. ACM CCS'93, pp. 62–73.
6. Bellare, M., Rogaway, P.: Optimal asymmetric encryption – How to encrypt with RSA. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, Springer, Heidelberg (1995)
7. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, Springer, Heidelberg (1996)
8. Bellovin, S., Merritt, M.: Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attack. In: Proc. IEEE Security and Privacy, IEEE Computer Society Press, Los Alamitos (1992)
9. Boneh, D., Ding, X., Tsudik, G., Wong, C., Method, A.: for Fast Revocation of Public Key Certificates and Security Capabilities. In: Proc. Usenix Security Symposium (2001)
10. Boyd, C.: Digital Multisignatures. In: Beker, H.J., Piper, F.C. (eds.) Cryptography and Coding, pp. 241–246. Clarendon Press (1989)
11. Boyko, V., MacKenzie, P., Patel, S.: Provably Secure Password Authentication and Key Exchange Using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, Springer, Heidelberg (2000)
12. Cramer, R., Shoup, V.: A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, Springer, Heidelberg (1998)
13. Dean, D., Berson, T., Franklin, M., Smetters, D., Spreitzer, M.: Cryptography as a Network Service.In: Proc. NDSS'01 (2001)
14. Denning, D.E.: Digital Signature with RSA and other Public-Key Cryptosystems. C. ACM 27(4), 388–392 (1984)
15. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong Key-Insulated Signature Schemes. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, Springer, Heidelberg (2002)
16. Ganesan, R.,Yaksha: Augmenting Kerberos with Public Key Cryptography. In: Proc. NDSS'95 (1995)
17. Goldreich, O., Goldwasser, S., Micali, S.: How to Construct Random Functions. J. ACM 33(4), 210–217 (1986)
18. Goldwasser, S., Micali, S., Rivest, R.: A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. SIAM J. Computing 17(2), 281–308 (1988)
19. Itkis, G., Reyzin, L.: Forward-Secure Signatures with Optimal Signing and Verifying. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, Springer, Heidelberg (2001)
20. Itkis, G., Reyzin, L.: SiBIR: Signer-Base Intrusion-Resilient Signatures. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, Springer, Heidelberg (2002)
21. Katz, J., Ostrovsky, R., Yung, M.: Efficient Password-Authenticated Key Exchange Using Human-Memorizable Passwords. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, Springer, Heidelberg (2001)
22. MacKenzie, P., Reiter, M.: Networked Cryptographic Devices Resilient to Capture. In: Proc. IEEE Security and Privacy, IEEE Computer Society Press, Los Alamitos (2001)
23. Matsumoto, T., Kato, K., Imai, H.: Speeding Up Secret Computations with Insecure Auxiliary Devices. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, Springer, Heidelberg (1990)
24. Perlman, R., Kaufman, C.: Secure Password-based Protocol for Downloading a Private Key. In: Proc. NDSS'99 (1999)

25. Pinkas, B., Sander, T.: Securing Passwords Against Dictionary Attacks. In: Proc. ACM CCS'02 (2002)
26. Rackoff, C., Simon, D.: Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, Springer, Heidelberg (1992)
27. Rivest, R.A., Shamir, A., Adleman, L., Method, A.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. C. ACM 21(2), 120–126 (1978)
28. Schneider, F.: Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial. ACM Comput. Surv. 22(4), 299–319 (1990)
29. Xu, S., Sandhu, R.: Two Efficient and Provably Secure Schemes for Server-Assisted Threshold Signatures. In: Proc. RSA Con. – Cryptographer's Track (2003)
30. Xu, S., Sandhu, R.: A Scalable Secure Cryptographic Service. Full version of the present paper, available at www.cs.utsa.edu/~shxu

gVault: A Gmail Based Cryptographic Network File System

Ravi Chandra Jammalamadaka[1], Roberto Gamboni[3], Sharad Mehrotra[1], Kent E. Seamons[2], and Nalini Venkatasubramanian[1]

[1] University of California, Irvine
[2] Brigham Young University
[3] University of Bologna, Italy
{rjammala, sharad, nalini}@ics.uci.edu, seamons@cs.byu.edu, roberto.gamboni@studio.unibo.it

Abstract. In this paper, we present the design of gVault, a cryptographic network file system that utilizes the data storage provided by Gmail's web-based email service. Such a file system effectively provides users with an easily accessible free network drive on the Internet. gVault provides numerous benefits to the users, including: a) Secure remote access: Users can access their data securely from any machine connected to the Internet; b) Availability: The data is available 24/7; and c) Storage capacity: Gmail provides a large amount of storage space to each user. In this paper, we address the challenges in design and implementation of gVault. gVault is fundamentally designed keeping an average user in mind. We introduce a novel encrypted storage model and key management techniques that ensure data confidentiality and integrity. An initial prototype of gVault is implemented to evaluate the feasibility of such a system. Our experiments indicate that the additional cost of security is negligible in comparison to the cost of data transfer.

1 Introduction

Network file systems have become quite popular in the past two decades. In such systems, user data in the form of files is stored at a remote server. The server is then in charge of providing services such as backup, recovery, storage, access, etc, thereby absolving the user from it's responsibility. The user can then *mount* the remote file system as a *local drive* and proceed to perform all the required file operations on the remote data. The biggest advantages of network file systems is that they allow users to remote access their data. A nomadic user can connect to the remote server from any machine connected to the Internet and access his information.

A related trend is the rise in popularity of web based email service providers (WESPs). Such services provide the users with the facility to send/receive emails free of charge. The business model is typically based on advertisements that are displayed on webpages the user is currently accessing. A big advantage of such systems, like network file systems, is that they allow the user to access his email from any machine connected to the Internet. Typically, WESPs allocate a lot of storage to the user to store his/her emails, which emphasizes the fact that online storage has become very cheap.

Imagine a network file system built over the storage offered by the WESPs. Such a file system has numerous advantages which include: a) Remote access: The users can access their data from any machine connected to the Internet; b) Availability: The data is available 24/7; and c) Storage capacity: The WESPs provide a large amount of storage space to the user. Such file systems already exist. GMail Drive [1] and RoamDrive [2] are examples of applications that layer a file system over the storage space provided by Gmail, a prominent WESP on the Internet. Google's usage policy [4] does not prevent such file systems to utilize their system for data storage.

The biggest drawback of the current systems is the inherent *trust* they place on the WESPs. The data is stored in plain text at the WESP and is vulnerable for the following attacks:

– **Outsider attacks:** There is always a possibility of Internet thieves/hackers breaking into the WESP's system and stealing or corrupting the user's data.
– **Insider attacks:** Malicious employees of the WESPs can steal the data themselves and profit from it. There is no guarantee that the confidentiality and integrity of the user's data are preserved at the server side. Recent reports indicate that the majority of the attacks are insider attacks [11].

There also have been instances of WESPs collaborating with repressive regimes and providing them personal data that belongs to users [5]. Fear of prosecution is in itself a valid reason not to trust WESP with our personal data. As such, use of WESPs for storing potentially sensitive information has been heavily limited.

In this paper, we address the problem of designing a cryptographic network file system called gVault, that utilizes the storage space provided by the WESP and addresses the drawbacks of the earlier systems that we previously mentioned. gVault runs on top of the Gmail's email service and provides the users a file system like interface. gVault is fundamentally designed to be used by an average computer user and does not trust the Gmail storage servers with his/her data. The reader should note that while we have designed and implemented a system over Gmail, the techniques that we develop in this paper are applicable to any Internet based storage provider. We choose Gmail since it is widely popular and free to use.

There are many challenges that need to be addressed in designing a system of this kind. The first set of challenges occur due to the requirement of *designing a file system that is easy to use.* Therefore, we made a fundamental decision to control the overall security that is offered by gVault by a master password, a secret known only to the user. Passwords can be forgotten or stolen, therefore we need techniques that can defend against such situations. The second set of challenges occur due to users requiring their data remain confidential. In order to ensure that no unauthorized recipients obtain a user's data, data needs to be encrypted. This further raises many questions: a) What is the granularity of encryption? b) How is key management done? c) What is the encrypted storage model at the Gmail side? Such a model should optimize the data storage and fetching operations at the server. The third set of challenges occur due to the *data integrity* requirements at the client side. Techniques/mechanisms should be in place that detect any data tampering attempts at the server side. The fourth set of challenges are *implementation* based. All the client side file operations need to be translated into HTTP requests that Gmail servers can understand.

Overview of our approach: gVault runs as an application on the user's machine. gVault prompts the user to enter his Gmail username, Gmail password and the master password. Once the users enters the information, gVault opens a session with the Gmail server and functions as an HTTP client. All the file operations that the user performs at the client side is mapped to their equivalent HTTP requests. The responses from the Gmail servers, which are encapsulated in HTML, are parsed to recover the required user's data. gVault implements the necessary cryptographic techniques to ensure the security of user data in the translation of file operations to HTTP requests. In our model, outside the security perimeter of client machine everything else is untrusted. We do not place any restrictions on the kind of attacks that can be launched.

Paper Organization: The rest of the paper is organized as follows: In section 2, we describe our encrypted storage model that provides data confidentiality. In section 2.1, we describe the relevant operations in the encrypted storage model. In section 3, we describe our techniques that allow the user to detect any tampering attempts at the server side. In section 4, we describe the implementation details of gVault. In section 5 and 6, we present the related work and conclude with future directions.

2 Encrypted Storage Model (ESM)

This section describes how a user's file system is represented at the Gmail servers. Our objective is to design and implement an encrypted storage model that preserves the *confidentiality* of user's data. More concretely, we want to design an encrypted storage model that does not reveal: a) *Content of the files*; b) *Metadata of the file system*; and c) *Structure of the file system* to the server. It is obvious that the file content must be protected. We believe that mere encryption of the files is insufficient in itself. Both the metadata and the structure of the file system also contain a lot of information about user's data and therefore it makes sense to hide them as well. Metadata of the file system contains information such as file names, directory names, etc., and file system's structure reveals information about the number of directories, the number of files underneath a directory, etc. If the storage model at the server side reveals the file structure an adversary can launch known plaintext attacks and discover information about the user's data.

Current network file systems models do not satisfy our requirements as they tend to reveal the structure of the file system to the server. For instance, the

```
Input: User's File System F
BEGIN
1. Decouple F into a File Structure
$F_S$ and $S_F = \{ F_1, \ldots F_n \}$ a set of files.
2. Encrypt the File structure $F_S$ and store it
at the server.
3. For every file $f_i$ in $S_F$
    Generate the object encryption key $k_i$
    Encrypt the file $f_i$ with $k_i$
    and store it at the server.
END
```

Fig. 1. Encrypted storage Model

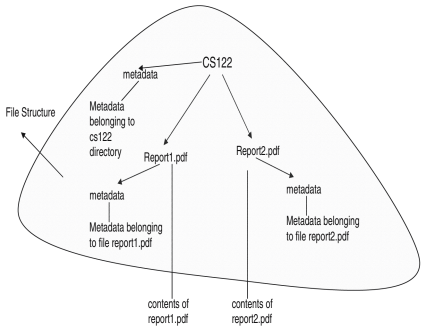

CS122
metadata
Metadata belonging to cs122 directory
File Structure
Report1.pdf
Report2.pdf
metadata
Metadata belonging to file report1.pdf
metadata
Metadata belonging to file report2.pdf
contents of report1.pdf
contents of report2.pdf

Fig. 2. Example of File Structure

NFS storage model [17] maps every file and directory at the client side to a file at the server side, thereby revealing the structure of the file system.

The following defines our encrypted storage model:

Definition 1. ***File System:*** *A user's file system is represented as a graph* $G =< V, E >$*, where* V *is a set of nodes that represents both files and directories. The set* E *represents the set of edges between the nodes. Let the function* $parent(n_1)$ *represent the parent node of node* n_1*. If node* n *is the parent of node* n_1*, we present the relationship as follows:* $n \leftarrow parent(n_1)$*. The edge set* E *contains all the edges between any two nodes* n_1 *and* n_2*, where* $n_1 \leftarrow parent(n_2)$ *or vice versa. For every node* $n \in V$*,* $n.metadata$ *represents the metadata that is associated with the node* n *and* $n.content$ *represents the content of the node.*

For a file node, its metadata is the name of the file, size of the file and last modified time. For a directory node, its metadata is the name of the directory, number of files and directories underneath it, size of the directory and last modified date. The content of the directory node is set to null.

We could have modeled the file system as a tree. Then such a model does not take into account the *symbolic links* that are present in the file system. The graph structure allows the incorporation of the symbolic links in the file system.

Definition 2. ***Server Side representation of the user's file system:*** *Let the user's file system* F *be a graph* $G =< V, E >$*. A user's file system* F *is represented as a tuple* $< F_S, S_F >$ *at the server , where* F_S *is the file structure of the file system and* S_F *is a set of file nodes* $\{F_1, F_2, \ldots F_n\}$ *that belong to the file system. The file structure* F_S *is a graph* $G^{'} =< V^{'}, E^{'} >$*, where* $V^{'}$ *contains all the nodes in* V *and* $E^{'}$ *contains all the edges in* E*. For all the file nodes* $n \in V^{'}$*,* $n.content$ *is a pointer to the relevant file in* S_F*.*

Fig 2 shows an example that illustrates the content of a file structure. We have decoupled the file system into a file structure object and a set of files. We will store these two data components separately at the server. As the server is untrusted, both F_S and S_F are encrypted before being stored at the server side. Fig [1] describes our overall mapping strategy to transform the plaintext at the client side to the ciphertext at the server side. For the rest of the paper, we will refer to both the file structure and individual files as data objects.

2.1 Operations in the ESM

Key generation: In gVault, the key to encrypt/decrypt the data object is generated on the fly depending on the metadata that is attached to the object. Using the metadata, gVault generates an object encryption key (OEK) for every object that is outsourced to the server.

We use the key derivation function (KDF) of the password based encryption specification PKCS #5 [8] to generate OEKs. The KDF function calculates keys from passwords in the following manner:

$$Key = KDF(Password, Salt, Iteration)$$

The *Salt* is a random string to prevent an attacker from simply precalculating keys for the most common passwords. The KDF function internally utilizes a hash function that computes the final key. To deter an attacker from launching a dictionary attack, the hash function is applied repeatedly on the output *Iteration* times. This ensures that for every attempt in a dictionary attack, the adversary has to spend a significant amount of time.

In gVault, the OEK K_{fs} for the file structure object is calculated as follows:

$$K_{fs} = KDF(MasterPassword, Salt, Iteration)$$

The OEK K_i for each file object is calculated as follows:

$$K_i = KDF(FileName||MasterPassword, Salt, Iteration)$$

The password paramater in the KDF function is the concatenation of the filename and the master password. Salt is a large random string that is generated the first time the object is created and is stored in plaintext along with the encrypted object. The filename does not include the full path name to permit a file objects to be moved between directories without changing its OEK. The iteration count is set to 10000, the recommended number.

The primary reason that we generate a key for every individual data object is to prevent cryptanalysis attacks, whose effectiveness increases with the amount of ciphertext available that is encrypted with the same key. Another approach is to generate a random key for each object and encrypt the object with that key. The random key could then be encrypted with the key derived from the password. We chose to generate the key since the KDF function is inexpensive compared to retrieving a key along with each object from the server. This reduces network bandwidth requirements, especially for small objects when the cost of retrieving the key would dominate.

Updating the file structure: The biggest drawback with our solution of decoupling of the file structure is that updates are not easy to handle. If the user makes changes to the file structure, we need to update the file structure stored at the server side. Updating the complete file structure for every update will reduce the performance of the system and make the user wait for a relatively long time.

gVault utilizes a novel approach similar in spirit to the approaches in journaling file systems to combat the above issues. gVault maintain a log of all the updates that take place in a session. Let L represent the log which is a set of log entries $\{L_1, \ldots L_n\}$. Each log entry L_i represents an update operation on the file system's structure. Update operations include cut, copy, paste, create and rename. For brevity, we will not describe the language used for representing such update operations. Whenever an update takes place, an appropriate log entry is added to the log and the file structure is updated locally. Let us assume that a user by issuing updates represented by the log L, has transformed the initial file structure F to $F^{'}$. When the user decides to log off, gVault encrypts $F^{'}$ and updates the file structure stored at the server side. After it successfully updates the file structure, gVault removes the log. If the application running at the client crashes, due to power failure or a hardware failure, etc., the gVault upon restart will look at the log L and reconstruct the file structure $F^{'}$ from F the last committed copy.

Notice that we primarily maintain the log for updates on the file structure. For updates which involve file transfers, we do not actually store the file contents on the log. Rather, a message is inserted into the log stating that a file transfer operation on particular file has started. When the file transfer is finished, a log entry is added to state the same. If due to a system crash, a file transfer is stopped midway, gVault will look at the log and figure out the failed operation and alert the user. It is up to the user now to take appropriate action.

Master password management: It is of paramount importance that a user does not reveal his/her master password to anyone. The loss of a master password could lead to disastrous effects, since now the adversary can have complete control over the user's file system. Passwords are prone to be lost or stolen. Therefore, there is a requirement to build mechanisms that change and recover the master password.

Changing the master password: Changing the master password will have its effect on the generation of keys. Hence, changing the master password could potentially be a very expensive operation, since it will require all the files/file structure to be decrypted with the old keys and encrypted again with the new keys. Fortunately, we can do this expensive operation *lazily.* gVault keeps track of both the new master password and the old master password in a configuration file that is encrypted with a key that is generated by the new master password. gVault will continue to decrypt the files that are fetched from the server with the keys generated from the old password. When a update is made to the file, then gVault will encrypt the updated file with the key that is generated with the new password. GVault will then set a flag in file structure to indicate that in future, it needs to use the key generated with the new master password. After a while, all the files will be encrypted with keys from the new master password. Lazily changing the master passwords is typically done to periodically update the master password, a recommended practice.

In a situation where the master password is compromised, the user could request the change of keys immediately. This is an expensive operation, the user has no choice but to wait till all the files are fetched, decrypted with the old key and encrypted with the new key. If the user decides to change the master password more than once, gVault will again force the change of all the keys.

Recovering the master password: We have designed a novel approach to recover the master password in case the user forgets it. Unlike traditional password recovery mechanisms, in our case the master password is also not known to the server. When the user first utilizes the gVault service, the application prompts the user to enter a set of question/answer pairs. Let Q_a represent such a set, where an element $Q_i \in Q_a$, is a tuple $< \mathcal{Q}, \mathcal{A} >$ where $\mathcal{Q}$ represents the question and $\mathcal{A}$ represents the answer. The user can enter any arbitrary number of questions and answers. In other words, the user can control the cardinality of set $|Q_a|$. In our implementation, gVault suggests a few questions, but it up to the user to select some of the questions or come up with his own. After the user selects the questions, he/she will then proceed to submit the relevant answers to the questions. The answers to these questions will be kept secret from the server, while the questions are stored in plaintext. A user needs to be careful in his/her choice of questions. The answers to these questions should typically lead to information that only the user knows about. An example of a suitable question is *"what is my social security number"*. The user is assumed never to forget the answers to these questions. We derive an encryption key called the *Recovery key* from the answers to these questions. The recovery key is derived as follows:

$$Recovery\_key = KDF(||Q_a, Salt, Iteration)$$

where KDF is the key derivation function discussed earlier. $||Q_a$ represents the concatenation of all the answers in the set Q_a. *Salt* is a random string that is stored in plaintext and iteration number is set to 1000. Now using the recovery key we compute *recovery information* RI of the master password as follows:

$$RI = E^k_{Recovery\_Key}(MP)$$

where MP refers to the master password and E^k represents the encryption function applied iteratively k times. Typically k is set to a value which increases the encryption time to the order of seconds. This is done to reduce the effectiveness of the brute force attacks. The adversary now also has to spend considerably more time per brute force attempt. RI is then stored at the server. When the user forgets his master password, gVault will fetch Q_A from the server and present the user with the questions. Note, we assume the user remembers his Gmail's username and password. After the user answers the questions, the *Recovery_key* is recalculated and master password is recovered as follows:

$$MP = D^k_{Recovery\_Key}(RI)$$

where D^k is the decryption function applied iteratively k times. Note that if the user does not provide the right answers, the recovered master password will not be the same as the original. The user can manually verify if that is the case, and can repeat the process if necessary. This process has similarities to the authentication protocols used by current websites, where an answer to a secret questions allows the user to recover a forgotten password. Therefore, the users are already familiar with such recovery mechanisms.

The password recovery is a useful feature, but it has the potential to be very insecure. Since the questions are in plain text, anyone with access to the server can learn the questions and launch a dictionary attack. This feature is only as secure as someone is able to make their question set strong against dictionary attacks. gVault provides a set of reasonable questions, the user would do well to select enough questions from this set to thwart a dictionary attack.

2.2 Analysis

Security: An adversary at the server side, by looking at the ciphertext stored at the server can procure some information regarding the file system of the user. The information includes: a) *The number of files and their relative sizes:* The size of the ciphertext dictates the size of the plaintext files; b) *The size of the file structure:* File structure is the first data object that is downloaded by the user upon login. The size of the file structure linearly increases with the number of nodes inside it. Hence, the adversary can reasonably guess the number of directories and files the user's file system contains. But, the adversary cannot find any information regarding the general hierarchy of the file system.

Another alternative for the encrypted storage model is to create a data object that subsumes both the file structure and the files. Such an object can then be downloaded at the beginning of the session. This representation provides more security, since the adversary does not know if the user has large number of files, or a large number of internal nodes in the file structure. Downloading the entire file system at the time of login puts an enormous performance strain on the system thereby making the system inherently not usable. The current design of the encrypted storage model strikes an appropriate balance between performance and security .

Performance: The file structure is fetched at the beginning of every session so that gVault does not need to contact the server while the user is navigating the file system. Adopting a storage model similar to the NFS model would force gVault to retrieve internal nodes on demand by visiting the server every time the user descends or ascends a level in the file structure. For slow Internet connections, the system will be less usable due to network latency at each navigation step. An improvement in the NFS model would be to allow the client to prefetch all of the internal nodes. Also, when the entire file structure is retrieved, the cost of decrypting the nodes individually is higher compared to a single bulk decryption of the complete file structure, since most encryption algorithms have a constant startup time. Thus, gVault encrypts the entire file structure as a single object.

It is possible that the user could pay a huge startup cost for downloading the complete file structure. The reader should note that this is a one time cost and in practice we have noticed that even for huge file systems, the file structure can be loaded quickly since its size tends to be very small.

Search: Another side benefit of the model is that it allows gVault to answer certain search queries. Since the complete file structure is cached locally, a client's queries on file and directory names can be executed locally without contacting the server. In order to conduct searches over the file content at the untrusted server, gVault would need to support searching over encrypted data. This is part of our future work, and we will explore existing solutions [7,9] and determine how they can be adapted to gVault.

3 Data Integrity

Another requirement of gVault is that data integrity be preserved. This section describes how gVault ensures the *Soundness* and *Completeness* of a user's data.

Soundness: To ensure soundness of a data object, gVault needs a mechanism to detect when tampering occurs. To achieve this, the HMAC[1] of an object is calculated and stored in the file structure along with the object/file node. When the object is retrieved from the server, its HMAC is also returned. The client calculates an HMAC again and compares it to the original HMAC. If they are equivalent, then no tampering has occurred. One way to compute the HMAC of an object O is as follows:

$$HMAC(O.Content||O.metadata)$$

Although the HMAC can be used to determine soundness, it does not guarantee the *freshness* of the object. That is, the server could return an older version of the object and the client will fail to detect it. One way to address this is to include the current version of the object when generating the HMAC. Thus, the HMAC can be generated as follows:

$$HMAC(O.id||O.Content||O.metadata||Version)$$

Every time the object is updated, the version number is incremented and the HMAC recalculated. This is done at the client side and hence there is no loss of security. In our model, the user is roaming and we do not store any data locally.

[1] A keyed-hash message authentication code.

Therefore, the impetus is on the user to validate the version number. Doing so, for every object he/she accesses will obviously make the system unusable. Fortunately, we can store the version numbers of all the file and directory nodes as metadata within the nodes themselves. That is, the version numbers are stored in the file structure. When the user accesses a file, gVault computes the HMAC of the file by using the version stored in the file structure and verifies whether it matches the HMAC stored in the file structure. If it matches, then it confirms that the user has the right version. While such an approach will work, there is still the problem of validating the version number of the file structure. In gVault, the last modified date is used as the version number for the file structure. The user is now entrusted with verifying the version number of the file structure. The user needs to do it only once at the beginning of every session and it is the only version number the user must validate. This is by no means a complete solution since it requires some manual validation and a human is known to be the weakest link in security architectures, but it builds some defenses in detecting if data tampering has taken place at the server.

Another method is to calculate the *global signature* of the complete file system using a Merkle tree approach [3] and store the signature locally. Whenever access to an object is made, the server sends a partial signature over the remaining objects so that the client can use the partial signature and the object being accessed to generate a signature to compare to the most recent global signature that is stored locally. If the signatures match, then no tampering has occurred. We did not adopt this solution because it requires server-side support, and violates our goal to use existing WESPs. Also, it requires a mobile user to transfer the global signature between machines, thereby pushing data management tasks back to the user, something that we want to avoid. An open problem is to design data integrity techniques that allows the client application to detect data tampering attempts at the server, without any user involvement.

Completeness: In gVault completeness is trivially achieved, since a client always knows the correct number of objects that need to be returned by the server. For instance, if the user is accessing a file, the server is required to return only one object.

4 gVault Prototype and Evaluation

This section describes the implementation details of the current gVault prototype. gVault is implemented in Java mainly for software protability reasons. gVault executes totally on the client side. No server side modifications are required. The current implementation of gVault is an executable jar file and hence does not require any installation. In the future, we will release gVault as an applet so that it can run from a web browser. Fig 3 illustrates the overall software architecture.

Components: The clients interact with gVault through the File System Local and the File System Remote components. The File System Local component

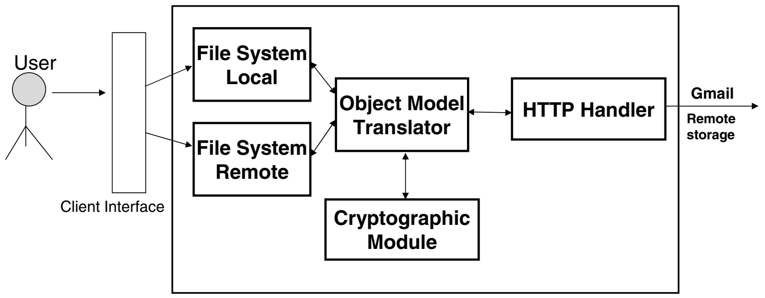


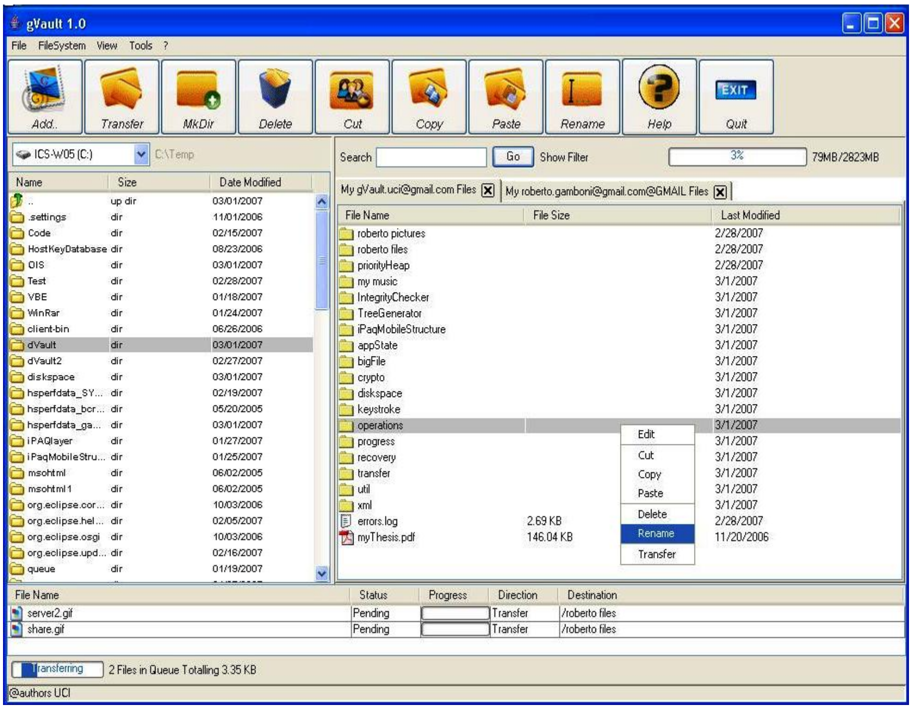
Fig. 3. gVault Software Architecture

provides a GUI interface to the local file system where the application is running. The File System Remote component provides a GUI interface to the file system stored at the remote server. The interface (see fig 4) of both these components is similar to the interfaces that exist to any modern file system. The Object Model Translator maps the file system the user is outsourcing into data objects. The Cryptographic Module supports the object model translator in the cryptographic operations. The HTTP Handler translates file operations into HTTP operations that Gmail servers support.

User Interface and Functionality: Fig 4 illustrates the screenshot of the gVault system. Users of gVault must establish a username and password with the Gmail service prior to using gVault. During login, the user has to submit the username/password of Gmail and the gVault master password that is used to control the cryptographic operations. Obviously the master password must be strong, since the security that gVault provides depends on the master password. gVault allows the users to maintain multiple accounts thereby realizing multiple file systems. One of these accounts is designated as the *primary account* and this account stores the required information that will allow gVault to open all the file systems, when the user provides the credentials of the primary account. The different remote file systems are shown in different tabs and the user can switch from one file system to another with a simple click.

gVault supports prioritized execution of basic file operations for better interactive response times. The HTTP handler maintains two priority queues namely the *Operation Queue* and the *File Transfer Queue*. The HTTP handler utilizes these two queues to schedule the file operations issued by the user. The operation queue is primarily used to queue operations to which the user expects immediate response. Examples of operations include, deleting a file, opening a relatively small file, etc. The File transfer queue maintains operations which the user should expect a longer response time such as opening a large file. The HTTP handler opens two sessions with Gmail simultaneously to schedule the operations from each of the queues separately.

Another important feature of gVault is its ability to handle large files. Currently, the atomic unit of storage is a file. The Gmail service has a limit on the



Name	Size	Date Modified
..	up dir	03/01/2007
.settings	dir	11/01/2006
Code	dir	02/15/2007
HostKeyDatabase	dir	08/23/2006
OIS	dir	03/01/2007
Test	dir	02/28/2007
VBE	dir	01/18/2007
WinRar	dir	01/24/2007
client-bin	dir	06/26/2006
dVault	dir	03/01/2007
dVault2	dir	02/27/2007
diskspace	dir	03/01/2007
hsperfdata_SY...	dir	02/19/2007
hsperfdata_bcr...	dir	05/20/2005
hsperfdata_ga...	dir	03/01/2007
iPAQlayer	dir	01/27/2007
iPaqMobileStru...	dir	01/25/2007
msohtml	dir	06/02/2005
msohtml1	dir	06/02/2005
org.eclipse.cor...	dir	10/03/2006
org.eclipse.hel...	dir	02/05/2007
org.eclipse.osgi	dir	10/03/2006
org.eclipse.upd...	dir	02/16/2007
queue	dir	01/19/2007

File Name	File Size	Last Modified
roberto pictures		2/28/2007
roberto files		2/28/2007
priorityHeap		2/28/2007
my music		3/1/2007
IntegrityChecker		3/1/2007
TreeGenerator		3/1/2007
iPaqMobileStructure		3/1/2007
appState		3/1/2007
bigFile		3/1/2007
crypto		3/1/2007
diskspace		3/1/2007
keystroke		3/1/2007
operations		3/1/2007
progress		3/1/2007
recovery		3/1/2007
transfer		3/1/2007
util		3/1/2007
xml		3/1/2007
errors.log	2.69 KB	2/28/2007
myThesis.pdf	146.04 KB	11/20/2006

File Name	Status	Progress	Direction	Destination
server2.gif	Pending		Transfer	/roberto files
share.gif	Pending		Transfer	/roberto files

2 Files in Queue Totalling 3.35 KB

Fig. 4. gVault Screenshot

maximum size of the file that can be outsourced. When a local file exceeds the limit, gVault transparently breaks a large file into a set of objects that conform to the Gmail's limitations. When access to the file is required, gVault fetches all the required objects and combines them for the user.

gVault supports the standard set of file operations. A user can create/open files and directories, delete files/directories, etc. We will now explain some of the implementation details for these operations.

Creating a file: To create a file, gVault first adds a log entry that includes the name of the file. Then, gVault adds the file to the local file structure. The file is then encrypted using its encryption key and stored at the server. To achieve this, gVault first calculates the file id by hashing the filename and the random salt generated during the key generation. Then, gVault creates an HTTP POST message that sends an email message to the user's Gmail account. The subject of this email contains the file id and the body of this email contains the encrypted file content.

Opening a file: To open a file, gVault must locate the corresponding email message containing the file. To accomplish this, gVault first calculates the file id. Then, gVault uses Gmail's search interface to retrieve the email according to the required file id in it's subject header. This requires that one HTTP POST request containing the search query (i.e., file id) be sent to the Gmail server. After gVault identifies the relevant email, it issues another HTTP GET message to retrieve the email. Once the email is fetched, gVault parses the body of the email, decrypts the file content and displays it to the user. Note that file creation requires one HTTP request, while opening a file requires two HTTP requests. Since gVault does not control how emails are stored at the Gmail server, it must search for an email message containing the desired file. .

Updating a file: To update a file, the client follows a similar pattern to creating a file. To create a file, the client needs to add a node to the file structure locally. To update a file, the corresponding node already exists locally, so gVault encrypts the content and stores it on the server.

Moving a file: To move a file, gVault first adds the relevant update operation to the log at the server. In gVault, the combination of a file name and its random salt is unique. Therefore, the file encryption key does not change when an object moves. Using the pathname in the key generation process would force gVault to decrypt the object and encrypt it again with the new key. Under the current design, all gVault needs to do is to update the file structure.

Deleting a file: To delete a file, gVault first updates the log with the delete operation. Similar to opening of the file, gVault first identifies the email that contains the file by using Gmail's search facility, and then sends an HTTP POST message that deletes the email from the server. Then, gVault updates the file structure stored locally at the client.

Other operations: Additional operations such as moving a directory, renaming a file, deleting a directory, etc. are not described due to space limitations. The implementation of these operations follows a similar pattern to the operations described previously.

4.1 Performance

We conducted experiments to measure gVault's performance in encrypting/decrypting data objects, calculating data integrity information, prefetching the file structure, and the network delays for transferring files.

Our experiments were conducted on an Intel Celeron(R) 1.80 Ghz processor with 768 MB Ram client machine. For the experimental data, we used a local file system of one of the authors. This data contained a wide assortment of files such as video files, mp3 files, word documents, excel spread sheets, text files, etc. The file system was outsourced via the gVault application to Gmail storage servers.

Fig 5 describes the cryptographic costs associated with gVault's usage. The cryptographic costs includes the encryption costs, decryption costs, and the integrity costs. As expected, encryption and decryption costs are nearly the same and the cost of calculating integrity information is lower than that of the cryptographic costs. Fig 6 shows the network costs. Note that when compared to

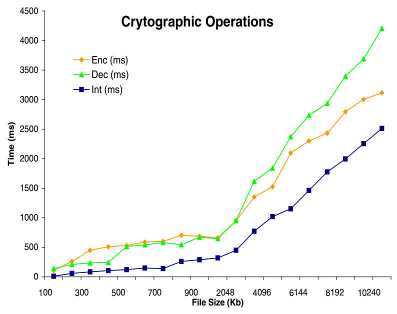


Fig. 5. Cryptographic operations

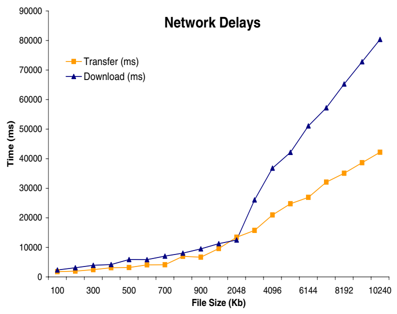


Fig. 6. Network delays in transferring files

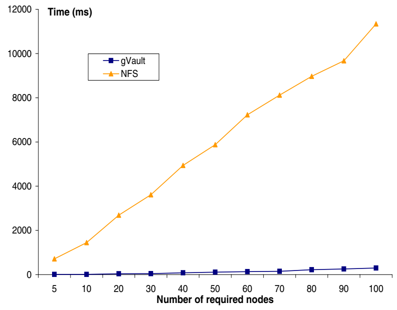


Fig. 7. Fetching the required nodes

network costs, cryptographic costs are nearly negligible. For instance, to transfer a 10 MB file to the Gmail server, it takes about 85 secs. The cryptographic costs for the same file totally amount to 6.5 secs.

We wanted to measure the performance gain due to prefetching the complete file structure. If we were to follow an NFS based storage model, then every time the user descends a level in the file system, the server needs to be contacted to fetch all the child nodes. In Gmail, there is no API available to directly download the emails that contain the required nodes. Therefore, gVault uses Gmail's search interface to find the required emails and individually download them. Fig 7 shows the comparison of this approach to the NFS approach. gVault does significantly better since it can fetch all the required nodes locally. In summary, we conclude that enabling secure storage over web-based data storage providers is feasible and cost effective.

5 Related Work

Cryptographic file systems [14,15,6] provide file management to users when the underlying storage is untrusted. This is typically the case when data is stored at remote untrusted servers. Cryptographic file systems can be classified under two categories: a) password based; and b) non-password based. Sirius [14] and Plutus [15] are examples of cryptographic file systems that are non-password based. Their goal is to provide the user with data confidentiality and integrity when the data is stored at a remote server. We differ from them in the following manner: a) these systems were built for sophisticated users. For instance, in Sirius and Plutus, the users are expected to purchase a public/private key pair and securely transport it when mobile access is desired. Our architecture is catered to average computer users and we placed heavy emphasis on making the system easy to use without sacrificing security.; and b) the encryption storage model leaked the file structure information which is not the case in gVault.

There are other cryptographic file systems that are password based. Most notable of them are the cryptographic file system(CFS) for Unix [6] and Transparent cryptographic file system(TCFS) for Unix [18]. CFS is a user level file system that allows users to encrypt at the directory level and the user is supposed to remember a pass phrase for every directory he/she intends to protect. TCFS is in many respects similar to CFS, except for the fact that cryptographic functions are made transparent to the user. To the best of our knowledge, both these system do not provide mechanisms to recover passwords. Besides the recovery mechanisms, gVault also differs from these systems in the storage model.

DAS [12,13] architectures allow clients to outsource structured databases to a service provider. The service provider now provides data management tasks to the client. The work on DAS architectures mainly concentrated on executing SQL queries over encrypted data. The clients of DAS architectures are mainly organizations that require database support. In this paper, our objective is to come up with a file system like service and hence we fundamentally differ from DAS related research works, even though we are similar in spirit.

6 Conclusions

This paper presented the design and implementation of gVault, a cryptographic network file system that provides a free network drive to the storage space offered by Gmail, a web-based email service. gVault protects the confidentiality and integrity of a user's data using cryptographic techniques. gVault provides users with a file system like interface and allows them to seamlessly access their data from any computer connected to the Internet.

gVault is designed for an average user. The overall security that gVault provides depends on the user remembering a master password. A mechanism is provided to change or recover the master password in case it is forgotten or stolen. This makes gVault usable for a wide spectrum of users. A beta version of gVault is available for download at http://gVault.ics.uci.edu.

Acknowledgements

We would like to acknowledge the work done by James Chou, Andrew Grant and Mark Lem who implemented parts of the gVault application. This research was supported by funding from the National Science Foundation under grant no. CCR-0325951 and IIS-0220069, the prime cooperative agreement no. IIS-0331707, and The Regents of the University of California.

References

1. Gmail Drive, http://www.viksoe.dk/code/gmail.htm
2. http://www.roamdrive.com
3. Merkle, R.: Protocols for public key cryptosystems. In: IEEE security and privacy, IEEE Computer Society Press, Los Alamitos (2000)
4. Gmail program policies, http://mail.google.com/mail/help/intl/en/program_policies.html
5. Man Jailed after Yahoo Handed Draft Email to China. http://www.ctv.ca/servlet/ArticleNews/story/CTVNews/20060419/yahoo_jail_ap_060419/20060419?hub=World
6. Blaze, M.: A cryptographic file system for UNIX. In: Proceedings of the 1st ACM conference on Computer and communications security, ACM Press, New York
7. Goh, E.j.: Secure Indexes (in submission)
8. RSA Laboraties. PKCS #5 V2.1: Password Based Cryptography Standard, ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-5v2/pkcs5v2_1.pdf
9. Song, D., Wagner, D., Perrig, A.: Practical Techniques for Searches on Encrypted Data. In: 2000 IEEE Symposium on Research in Security and Privacy, IEEE Computer Society Press, Los Alamitos (2000)
10. Britney, A.: The 2001 Information Security Industry Survey 2001 (cited, October 20, 2002), http://www.infosecuritymag.com/archives2001.shtml
11. Dhillon, G., Moores, S.: Computer crimes: theorizing about the enemy within. Computers & Security 20(8), 715–723
12. Hacigumus, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over Encrypted Data in the Database-Service-Provider Model. In: 2002 ACM SIGMOD Conference on Management of Data (June 2002)
13. Damiani, E., Vimercati, S.C., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational DBMSs. In: Proceedings of the 10th ACM conference on Computer and communications security, ACM Press, New York
14. Goh, E., Shacham, H., Modadugu, N., Boneh, D.: SiRiUS: Securing remote untrusted storage. In: Goh, E., Shacham, H., Modadugu, N., Boneh, D. (eds.) Proc. Network and Distributed Systems Security (NDSS) Symposium (2003)
15. Kallahalla, M., Riedel, E., Swaminathan, R., Wang, Q., Fu, K.: Plutus: Scalable secure file sharing on untrusted storage. In: Proc. 2nd USENIX Conference on File and Storage Technologies (FAST) (2003)
16. Zadok, E., Badulescu, I., Shender, A.: Cryptfs: A Stackable vnode level encryption file system. Technical Report, Columbia University (1998), CUCS-021-98
17. Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., Noveck, D.: NFS version 4 protocol. RFC 3530 (April 2003)
18. Cattaneo, A.D.S.G., Catuogno, L., Persiano, P.: Design and implementation of a transperant cryptographic file system for UNIX. In: FREENIX Track: 2001 Usenix annual technical conference (June 2001)

Design and Analysis of Querying Encrypted Data in Relational Databases

Mustafa Canim and Murat Kantarcioglu

Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083
{mxc054000, muratk}@utdallas.edu

Abstract. Security and privacy concerns as well as legal considerations force many companies to encrypt the sensitive data in databases. However, storing the data in an encrypted format entails non-negligible performance penalties while processing queries. In this paper, we address several design issues related to querying encrypted data in relational databases. Based on our experiments, we propose new and efficient techniques to reduce the cost of cryptographic operations while processing different types of queries. Our techniques enable us not only to overlap the cryptographic operations with the IO latencies but also to reduce the number of block cipher operations with the help of selective decryption capabilities.

1 Introduction

Sensitive data ranging from medical records to credit card information are increasingly being stored in databases and data warehouses. At the same time, there are increasing concerns related to security and the privacy of such stored data. For example, according to a recent New York Times article [1], records belonging to more than hundred million individuals have been leaked from databases in the last couple of years. Although currently criminals have not taken advantage of such disclosures effectively, there is an obvious need for better protection techniques for sensitive data in databases. Common techniques such as access controls or fire-walls do not provide enough security against hackers that use zero-day exploits or protection from insider attacks. Once a hacker gets an administrator access to a server that stores the critical data, he/she can easily bypass the database access control system and reach the entire database files. Although it brings some extra cost, encrypting the sensitive data is considered as an effective last line of defense, to counter against such attacks [2]. Also recently, legal considerations [3] and new legislations such as California's Database Security Breach Notification Act [4] require companies to encrypt sensitive data. To assist its customers with legislation compliance hurdles, Microsoft recently developed a new SQL server that comes with built in encryption support [5]. IBM also offers a similar functionality in its DB2 server, in which data is encrypted (and decrypted) using a row-level function [6]. Clearly, if the encryption keys are not compromised, a hacker (or a malicious employee) that controls the system will not be able to read all the sensitive data stored on the hard disk.

Cryptographic operations, required to store sensitive data securely, entail significant amount of cumbersome arithmetic calculations. Coupled with bad design choices, expensive cryptographic operations needed for querying and processing encrypted data could decrease the system performance dramatically. On the other hand, good design choices may drastically affect the overall performance. With this in mind, in this paper, we discuss some of the design issues to encrypt and query the sensitive data that resides in database disks.

To reduce the performance loss that arises from using cryptographic techniques, we analyzed different block cipher modes of operations and compared their performances under various disk access patterns to see which modes are suitable for databases and allow us to parallelize the IO latencies with the cryptographic operations. Based on our experiments and analyses, we suggest using an efficient and provably secure encryption mode of operation that also enables selective decryption of large data blocks. We also propose a new approach for storing and processing encrypted data and we show that by using suitable encryption mode, the additional time needed for processing different types of queries can be significantly reduced. Before describing our approach, we briefly discuss the threat model assumed in this paper.

1.1 Threat Model

In this paper, we assume a threat model similar to the one considered in [7], where the database server is trusted and only the disk is vulnerable to compromise.

We consider an adversary that can only see the files stored at the disk but not the access patterns. In this case, we just need to satisfy the security under chosen plain text attacks. In other words, we guarantee that (assuming used blockcipher (e.g. AES) is secure) by looking at the contents of the disk, any polynomial-time adversary will have negligible probability of learning anything about the sensitive data.

Specifically, we assume that (See Figure 1):

- *The storage system used by the database system is vulnerable to compromise.* Only the files stored in the storage system are accessible to the attacker.
- *Query Engine and Authentication Server is trusted.* We assume that queries are executed on a trusted query engine.
- *All the privacy-sensitive data will be stored encrypted.* From the previous work related to inference controls, we know that probability distribution information may create unintended inference channels. Since we do not know such inference channels in advance, we want our solution not to create any inference problem. Therefore, in addition to sensitive data, all the information that can reveal sensitive data (e.g. log files) will be stored encrypted.

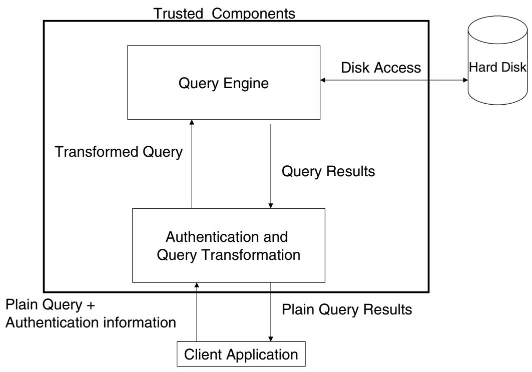


Fig. 1. Information flow and trust model for querying the encrypted data

1.2 Related Work

In [8], Bayer and Metzger explored the idea of using block ciphers and stream ciphers for encrypting B+ trees and random access files. They suggest generating different keys for each page based on the page id to break potential correlation attacks (pages encrypted with the same key and the same data will have the same cipher text). The disadvantage of this approach is, changing keys for each page imposes big key initialization costs. Hardjono and Seberry [9] suggested special combinatoric structures to disguise keys in B+ trees. Unfortunately, their combinatoric approach has not received the level of scrutiny required to achieve trust (as have others such as AES).

Querying encrypted data where even the database server is not trusted was first suggested in [10]. Hacigumus et al. [10] suggested partitioning the client's attribute domains into a set of intervals. The correspondence between intervals and the original values kept at the client site and encrypted tables with interval information are stored in the database. Efficient query of the data is made possible by mapping original range and equality query values to corresponding interval values. In subsequent work, Hore et al. [11] analyzed how to create the optimum intervals for minimum privacy loss and maximum efficiency. In [12], the potential attacks for interval based approaches were explored and models were developed to analyze the trade off between the efficiency and the disclosure risk. This line of work is different from our current problem because we assume that the database server is trusted and only the disk is untrusted.

In [7], Agrawal et al. suggested a method for order preserving encryption (OPES) for efficient range query processing on encrypted data. Unfortunately, OPES only works for numeric data and is not trivial to extend to discrete data.

In [13], Iyer et al. suggested data structures to store and process sensitive and non-sensitive data efficiently. The basic idea was to group encrypted attributes on one mini-page (or one part of the tuple) so that all encrypted attributes of a given table can be decrypted together. In [14], Elovici et al. also suggested a different way for tuple level encryption.

Our consideration is different from the ones in [13,14] in many aspects. Unlike the previous work, we provide:

– Complete analysis of block cipher modes suitable for databases,
– Analysis of the experiments about overlapping the IO latencies with the cryptographic operations by using multi-threading,
– A new approach for storing encrypted data in database pages by utilizing selective decryption property of CTR mode.

1.3 Organization of the Paper

In Section 2, we analyze the performance of different encryption modes under different granularity and disk access patterns. In Section 3, we discuss tuple level, page level, and mini page level encryption approaches and introduce a new approach for keeping records encrypted. Finally, we conclude the paper with the discussion of other related issues in querying encrypted data.

2 Block Cipher Modes Suitable for Databases

To store privacy-sensitive data securely, we need to use encryption methods secure against chosen plaintext attacks. Also we need general solutions that could support all kinds of data that needs to be encrypted. For example, the Order Preserving Encryption (OPES) idea suggested in [7] works for only numeric data.

Securely encrypting any kind of data is well studied in the cryptography domain. Any kind of long data is encrypted using operation modes based on secure block ciphers (e.g, AES[15]). All of these encryption modes process the long data by dividing into fixed size blocks(e.g 16 bytes in AES[15]) that can be processed by the block cipher. The obvious question is which block cipher mode is preferable for encrypting sensitive data in databases. To choose the best mode possible, we need to analyze the effect of these modes under specific database operation conditions. Specifically we look at:

– **Performance of Encryption Modes under Different Granularity:** We need to encrypt and decrypt the data at different granularity. If we are encrypting at the tuple level, we may need to decrypt small blocks at a time, if we use page level encryption, we need to decrypt potentially a page long encrypted data. Ideally, we should have a mode where different granularity has little effect on the total performance.
– **Performance of Encryption Modes under Different Disk Access Patterns:** We need to have an encryption method that enables efficient decryption under different disk access patterns.

First, in section 2.1, we give an overview of the block cipher modes that are suggested by the National Institute of Standards and Technology (NIST). Later on, in section 2.2 and 2.3, we analyze the performance of different block cipher modes under different encryption granularity and disk access patterns. Finally, we conclude this section with the discussion of which block cipher mode to choose for database encryption.

2.1 Overview of Block Cipher Modes

Before we briefly give an overview of different block cipher modes, we discuss the notations we use through out the paper. Let $E_K()$ be the encryption operation with the key K and $D_K()$ be the decryption operation with the key K. Let P_i be the i^{th} plain text block, C_i be the i^{th} ciphertext block, IV be the initial random vector and b be the block cipher input size (e.g 128 bits for AES[15]). Let $LSB_s(x)$ be the s least significant bits of x and similarly let $MSB_s(x)$ be the s most significant bits of x. Also below, $S[i..j]$ denotes the i^{th} through j^{th} bit of string S , $||$ denotes the string concatenation, and $\oplus$ denotes the bitwise xor operation. Table 1 summarizes the notations used for describing block cipher modes.

Table 1. Notations used for describing block cipher modes

$E_K()$	Block cipher encryption with the key K
$D_K()$	Block cipher decryption with the key K
b	Block cipher block length in bits
C_i	i^{th} ciphertext block
P_i	i^{th} plain text block
$LSB_s(x)$	s least significant bits of x
$MSB_s(x)$	s most significant bits of x
$S[i..j]$	i^{th} through j^{th} bit of string S
$\|\|$	String concatenation
$\oplus$	binary xor operation

Also, for the sake of simplicity, we assume that the plaintext P is n blocks long (i.e. $n \cdot b$ bit long plaintext). Under these assumption, the block cipher modes of operations suggested by NIST can be summarized as follows:(The details can be found in FIPS-SP 800-38A [16])

– **Electronic Code Book(ECB):** In ECB mode, each block of the plaintext is encrypted independently. Similarly each block of the ciphertext decrypted independently. Unfortunately, this mode is not secure since it reveals distribution information. (Note that if $P_i = P_j$ then $C_i = C_j$)

ECB Encryption: $C_i = E_K(P_i),\ i = 1..n$

ECB Decryption: $P_i = D_K(C_i),\ i = 1..n$

- **Cipher Block Chaining(CBC):** Each block of the ciphertext is created by encrypting the result of the previous ciphertext block xored with the current plaintext block.

  CBC Encryption:
  $C_1 = E_K(P_1 \oplus IV)$
  $C_i = E_K(P_i \oplus C_{i-1}),\ i = 2..n$

  CBC Decryption
  $P_1 = D_K(C_1) \oplus IV$
  $P_i = D_K(C_i) \oplus C_{i-1},\ i = 2..n$

- **The Cipher Feedback Mode:(CFB)** In this mode, successive segments (let s be the size of these segments where $1 \leq s \leq b$ and C_i^s, P_i^s denote the s-bit sized segments of ciphertext and plaintext respectively.) of ciphertext are concatenated to create an input for forward cipher to create blocks that are xored with plaintext segments.

  CFB Encryption:
  $I_1 = IV$
  $I_i = LSB_{b-s}(I_{i-1})|C_{i-1}^s, i = 2..n$
  $O_i = E_K(I_i), \quad i = 1..n$
  $C_i^s = P_i^s \oplus MSB_s(O_i), \quad i = 1..n$

  CFB Decryption:
  $I_1 = IV$
  $I_i = LSB_{b-s}(I_{i-1})|C_{i-1}^s, i = 2..n$
  $O_i = E_K(I_i), \quad i = 1..n$
  $P_i^s = C_i^s \oplus MSB_s(O_i), \quad i = 1..n$

- **Output Feedback Mode(OFB):** This mode is very similar to CFB with s taken as b. In this simplified description of OFB, we assume that ciphertext is $n * b$ bits long.

  OFB Encryption:
  $I_1 = IV$
  $I_i = O_{i-1}, \quad i = 2..n$
  $O_i = E_K(I_i), \quad i = 1..n$
  $C_i = P_i \oplus O_i, \quad i = 1..n$

  OFB Decryption:
  $I_1 = IV$
  $I_i = O_{i-1}, \quad i = 2..n$
  $O_i = E_K(I_i), \quad i = 1..n$
  $P_i = C_i \oplus O_i, \quad i = 1..n$

- **Counter Mode(CTR):** In this mode a counter(ctr) is incremented and the encrypted counter value is xored with plaintext block to get the ciphertext block.

  CTR Encryption:
  $C_i = P_i \oplus E_K(ctr + i), i = 0..n-1$

  CTR Decryption:
  $P_i = C_i \oplus E_K(ctr + i), i = 0..n-1$

2.2 Evaluating the Performance of Encryption Modes Under Different Encryption Granularity

In [13], Iyer et al. states that encrypting the same amount of data using few encryption operations with large data blocks is more efficient than many operations with small data blocks. In order to support this claim, they conduct an experiment using three ciphers: AES[15], DES[17] and Blowfish[18] under different data blocks: 100 bytes, 120 bytes, 16 KBytes. Based on the results of this experiment, they conclude that encrypting data few small units at a time takes longer than encrypting the same amount of data using larger data blocks.

As they stated in their paper [13], the key initialization costs constitute the most important cause of the high amount of time spent in encrypting small blocks of data. We expect to observe less time difference when the data is encrypted under different granularity if we can use **one key** per database table. Using one key per database table is feasible for many practical applications, because we can encrypt 2^{128} bits of data securely [19] using CTR mode of encryption with a given key. To test this view, we conducted some experiments to investigate the effect of key initialization. These experiments are performed under different encryption modes to see which mode is more appropriate to use under different encryption granularity.

In our experiments we used the OpenSSL [20] crypto library's CBC, CTR, OFB and CFB implementation of AES[15]. We chose AES because it is the current standard and supported by various companies. ECB is not used since it is not regarded as a secure mode of operation. We also implemented a slightly modified version of CTR (referred as CTR4). Modified CTR implementation encrypts all the counter values first (i.e. calculates all the $E_K(ctr+i)$ first) and then xors it with the plain text data. Algorithms 1 and 2 describe the details of the modified CTR implementation. In Algorithm 1, we first create a string S using the encrypted counter values and then xor the first l bit of S with plaintext message P to get the ciphertext C. Since encrypted counter values are xored with the plaintext, we do not need to do any padding if the size of the plaintext is not the multiple of the block-cipher length b.

Algorithm 1. Modified CTR Encryption (CTR4)

```
Require: Plain text $P$ with length $l$, initial counter value $ctr$, and block-cipher length $b$.
  Set $S = E_K(ctr+0)||E_K(ctr+1)||\ldots||E_K(ctr+\lceil \frac{l}{b} \rceil - 1)$
  Set $C = P \oplus S[0..l-1]$
  return $(ctr, C)$
```

Similarly, in Algorithm 2, we first create a string S using the encrypted counter values and then xor the first l bit of S with ciphertext C to get the plaintext P

Algorithm 2. Modified CTR Decryption (CTR4)

```
Require: Ciphertext $C$ with length $l$, initial counter value $ctr$, and block-cipher length $b$.
  Set $S = E_K(ctr+0)||E_K(ctr+1)||\ldots||E_K(ctr+\lceil \frac{l}{b} \rceil - 1)$
  Return $P$ where $P = C \oplus S[0..l-1]$
```

Since all of the block cipher operations are done independent of ciphertext data, CTR4 decryption can be executed in a multi-threaded fashion. One thread could be used for creating the encrypted counters (i.e., for computing S in Algorithm 2), and other thread could be used to read the encrypted data from the disk (i.e. reading C from the disk in Algorithm 2). The original implementation of CTR mode in OpenSSL crypto library calculates C_{i-1} before calculating $E_K(ctr + i)$. Thus it is not suitable for multi-threading.

Experiments: All of our experiments are conducted on a 2.79 GHz Intel Pentium D machine with 2 GB memory on Windows XP platform. We ran each experiment five times and reported the average results.

In the first experiment, we encrypted a total of one GB data which is cached in memory 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192 bytes at a time. In this experiment, the same key is used for encrypting the entire one GB data, and the key initialization is done only once. In other words, AES_set_encrypt_key() function is called once. [1] In Table 2, we report the results of experiment 1 in seconds.

Table 2. Encryption of 1 GB data under different block sizes

	Experiment 1 (**With** key initialization)					Experiment 2 (**Without** key initialization)				
Block size (Byte)	CBC	CTR	OFB	CFB	CTR4	CBC	CTR	OFB	CFB	CTR4
16	24.98	26.19	27.09	25.64	23.02	49.27	50.77	50.50	50.84	47.91
64	24.31	25.75	26.52	25.03	22.81	30.44	32.13	31.41	31.53	29.16
128	23.78	25.56	26.27	25.20	22.31	26.92	28.83	28.27	28.58	25.44
256	23.56	25.50	26.39	25.44	22.11	25.17	27.42	26.61	27.09	23.95
512	23.42	25.50	26.48	25.41	22.17	24.34	26.52	25.89	26.22	23.23
1024	23.38	25.52	26.47	25.39	22.03	24.00	26.23	25.45	25.75	22.78
2048	23.33	25.48	26.55	25.42	22.05	23.78	25.98	25.23	25.66	22.63
4096	23.36	25.53	26.34	25.38	22.28	23.72	25.84	25.25	25.34	22.78
8192	23.25	25.56	26.39	25.41	22.30	23.53	25.84	25.14	25.42	22.75

In the second experiment, we again encrypted one GB of data similar to experiment 1 but in this experiment, key initialization is done at the beginning of each data block. Average results of experiment 2 are shown in Table 2 in seconds.

Experiment 1 indicates that CTR4 mode is the fastest, if the key initialization done just once at the beginning of the encryption process. Also, if we compare the results of experiment with or without key initialization for CTR4 (shown in figure 2), we can observe that encrypting larger blocks requires less amount of time if the key initialization is performed every time before encrypting each data block. However, the time required to encrypt the data does not depend on block size if key initialization is done just once. In other words, if the blocks used are not too small (i.e. larger or equal to 64 bytes) and one key is used per database table, the encryption block size does not effect the total decryption times. Actually, it

[1] Initialization means generating substitution tables based on the secret key and this can be a costly operation as reported in [13].

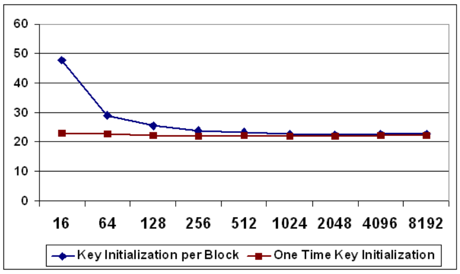


Fig. 2. Effect of key initialization under different granularity

is easy to see this fact from the API of modern crypto packages like the OpenSSL Library. For example in OpenSSL, to start encrypting with a key (similar for decryption), the key initialization function (AES_set_encrypt_key()) should be called first. After the initialization is done, encryption function can be called on various size blocks.

At the same time, experiment 1 indicates that encrypting small blocks (i.e. less than 64 bytes) at a time could be a problem. Fortunately, clever usage of CTR4 mode can solve the small block size problem. (This is not possible with other modes) Since each block is encrypted independently, we can combine blocks from different tuples (i.e., combine 4 blocks from 4 different tuples to create a 64 byte block) and decrypt/encrypt them together.

2.3 Performance of Encryption Modes Under Different Disk Access Patterns

In this part we conducted four experiments to evaluate the performance of encryption modes under random and sequential disk access patterns. In each experiment, data is accessed 4 KB page at a time, since it is the default page size in many DBMSs such as IBM DB2 [21]. We describe those four experiments below.

- **Experiment 3:**
  In the first part, one GB of encrypted file cached in memory is read **sequentially** and decrypted using different encryption modes. In the second part, the same experiment is repeated with the same file which is accessed from disk (i.e. file is not cached in memory). The results of the experiment are shown in Table 3.
  The results show that whether the data resides in memory or not, does not affect the time for reading and decrypting data sequentially. This implies that during the decryption of the current page disk controller can prefetch the next page.
- **Experiment 4:**
  In the first part, 512 MB of the 1 GB encrypted file is accessed **randomly** 4K page at a time from memory and decrypted using different encryption

Table 3. Results of experiment 3 (reading and decrypting 1 GB file sequentially)

Crypto Mode:	CBC	CTR	OFB	CFB	CTR4
from memory	29.02	31.17	30.92	30.89	27.89
from disk	28.91	31.94	30.5	31.05	27.31

Table 4. Results of experiment 4 (reading one GB file randomly and decrypting 512 MB)

Crypto Mode:	CBC	CTR	OFB	CFB	CTR4
from memory	15.7	16.73	16.38	16.52	15.03
from disk	262.63	262	271.94	266.3	267.64

modes. In the second part, the same experiment is repeated with the same file which is accessed from the disk. The results of the experiment are shown in Table 4

The results of the experiment 4 indicate that random access to the pages causes significant delays if the data is not cached in the memory.

– **Experiment 5:**
Multi-threaded version of CTR4 [2] is used to decrypt the 512 MB **randomly** accessed data, which took 257.3 seconds in the average. This experiment is not performed with the other modes since they do not allow multi threading during decryption.

If the result of experiment 5 is compared with experiment 4, we can see that multi threaded version of CTR4 reduces the decryption cost significantly, by overlapping the IO operations with decryption operations. In experiment 4, decrypting 512 MB of 1 GB file randomly takes 266.1 seconds in the average. However, performing the same experiment takes 257.3 seconds in this experiment. Therefore, multi threaded version of CTR4 runs almost 9 seconds faster than other modes of operations in the average.

In order to support this claim, we have calculated the time to decrypt 512 MB allocated memory space sequentially with CTR4, which took 11.2 seconds to perform. This result indicates that 9 seconds of running experiment 5 is overlapped with IO operations.

This implies that using hardware accelerators, most of the time required for cryptographic operations can be overlapped with the random access IO operations.

– **Experiment 6:**
Multi-threaded CTR4 is used to decrypt the 1 GB file **sequentially**. The average of the results indicates that it takes 28.05 seconds to decrypt the file sequentially. When compared with the results of experiment 3, it can be

[2] i.e One thread reads the data, other thread encrypts the counter values. When both threads are done, their results are xored to get the plain text.

concluded that the time to read and decrypt one GB file sequentially is not significantly reduced by using the multi threaded version of CTR.

The above results imply that the cost of the cryptographic operations can be overlapped with the cost of the IO operations using fast hardware based multi-threaded implementation of CTR mode, which could improve the performance especially in storage area networks where disk access latency could be higher.

2.4 Which Mode?

We would like to have a block cipher mode that is suitable for **efficient** processing of encrypted data in databases. Due to reasons stated below, CTR mode emerges as the **best** choice among classic and proven to be secure block cipher modes for efficiently querying encrypted data. The properties of CTR mode that is useful for database encryption can be given as follows:

- **Efficient Implementation:** In all of our experiments, modified version of CTR (CTR4) was the fastest. Since the encryption of each block is independent, modern processor architecture's properties such as aggressive pipelining, multiple cores, and large number of registers can be utilized for even more efficient implementation [22]. For example, in [22], optimized version of CTR mode is four times faster than the optimized version of CBC mode. Also CTR mode is suitable for parallel processing.
- **Selective Decryption:** CTR mode could be used to decrypt arbitrary parts of the plaintext. For example, for each tuple encrypted using CTR mode, during the selection operation, we may just decrypt the selection field first and decrypt the rest if the selection criteria is satisfied. To see why we can decrypt the arbitrary substring of a given ciphertext, consider the algorithm 3. In algorithm 3, first the counter values that are used to encrypt the $P[u..v]$ are calculated. Later on, using the counter values $\lfloor \frac{u}{b} \rfloor$ and $\lceil \frac{v}{b} \rceil$, encrypted counter values are created. Finally, the appropriate segment of the encrypted counter values are xored with the required part of the ciphertext (i.e. $C[u..v]$) to compute $P[u..v]$.

  Other block cipher modes such as CBC, CFB, OFB defined for AES[15] in the FIPS-SP 800-38A[16] standard do not allow for selective decryption because the encryption of a block depends on the encryption of the previous block. This implies that we cannot selectively decrypt the required part of the data. ECB and CTR modes of operation encrypt each block independently. Unfortunately, ECB mode reveals the underlying distribution of the data. Therefore, it does not provide the desired level of security.
- **Preprocessing:** In CTR mode, most costly part of the encryption and decryption (evaluating $E_K(ctr+i)$) can be done without seeing the data. Actually, we used this property in Experiment 5 and showed that this can reduce the encryption cost significantly. This is not possible for all the modes except OFB mode but OFB mode does not allow selective decryption.

Algorithm 3. Decryption of arbitrary substring of ciphertext in CTR mode

```
Require: Ciphertext $C$ with length $l$, initial counter value $ctr$, block-cipher length $b$,
    decryption start index $u$ and decryption end index $v$
    Set $S = E_K(ctr + \lfloor \frac{u}{b} \rfloor)|| \ldots ||E_K(ctr + \lceil \frac{v}{b} \rceil - 1)$
    return $P[u..v] = C[u..v] \oplus S[(u - \lfloor \frac{u}{b} \rfloor \cdot b)..(v - \lfloor \frac{u}{b} \rfloor \cdot b)]$
```

3 A New Approach for Storing Encrypted Data in Database Pages

Granularity of encryption is another important design issue that needs to be considered for efficient storage of encrypted data. The overall database performance can potentially be affected by small changes in the design choices regarding the way of keeping the records in the pages. Tuple level and page level encryption are the most well known options for this purpose. Alternatively, we can use the mini-page approach suggested in [13].

In tuple level encryption, each tuple is encrypted or decrypted separately. If the database needs to retrieve some of the tuples, there is no need to decrypt all of the tuples in the table.

Page level encryption will correspond to a mechanism where a particular page is completely decrypted when buffer pool needs to access the page from the disk. After some modifications, the page is encrypted again and written to disk.

Mini page level for database encryption was first suggested for encrypted data in [13] based on the work of Ailamaki et.al [23]. In this technique, when a tuple is inserted into a page, its attributes are partitioned and stored in corresponding mini pages on the same page.

Deciding to use one of these approaches depends on two factors:

- the cost of the required block cipher operations under different types of queries
- the amount of modification required to implement these approaches in conventional databases

The mini page level encryption, as discussed in [13], is designed in such a way that the sensitive attributes of the records are encrypted with AES [15] in CBC mode and inserted at the beginning of the page. When a page is read from disk, just the first part of the page is decrypted. Since all the sensitive attributes of the records are located at the beginning of the page, the decryption process continues until all sensitive attributes are decrypted.

Although this idea is a neat solution for encrypting sensitive data, there are two issues that are not addressed. The first issue is that it does not allow selective decryption since the encryption mode is selected as CBC mode. When a projection query needs to get specific attributes among the sensitive attributes, all of the encrypted attributes should be decrypted. This cost will linearly increase if the length of the sensitive attributes increases proportionally to the total length of the records.

Secondly, the mini page level encryption requires significant amount of modification in the page structure of conventional databases. This is because, the attributes of the records are kept in two different parts of the page rather than in a consecutive order.

In the page level encryption, the whole of the page is encrypted before writing into disk and decrypted before loading into buffer pool. If all the data needs to be kept secret, this level of encryption is appropriate. In addition to that, implementing such an approach requires less modification in the page structure of existing databases. However, this is not preferable since it unnecessarily encrypts nonsensitive data along with sensitive data.

Because of the problems discussed above, we propose a new encryption method which we refer to as page level CTR. This method basically utilizes the selective decryption property of CTR4 and combines the useful aspects of page level and tuple level encryption methods. Unlike the mini page approach, sensitive and nonsensitive attributes of the records are kept consecutively. The decryption is performed for each sensitive attribute of the records separately after a page is read from the disk. Therefore, it resembles to tuple level approach.

In our encryption method, we propose using CTR4 for cryptographic operations. So we need to somehow store the counter values in the page. Since each counter value requires a 16-byte of location, keeping one counter value for each record will entail a nonnegligible storage cost. Fortunately, we can use one counter value per page instead of one record. With respect to storage of counter values, our page level CTR approach is similar to page level encryption. Also we modified and optimized the increment function of CTR mode to increment counter values as much as needed at a time, instead of just one by one.

The page structure of the proposed method is illustrated in figure 3 with an example, where we assume that a projection query requires to read the encrypted attributes of two consecutive records. The start index of the attribute of record 1 is $\alpha_1 = 220$ bytes and of record 2 is $\alpha_2 = 430$ bytes, with respect to the beginning of the page. Before starting the decryption, the counter value of the page is read from the beginning of the page. Then this value is incremented by δ_1 where $\delta_1 = \lfloor \frac{\alpha_1}{\beta} \rfloor = \lfloor \frac{220}{16} \rfloor = 13$ with the optimized increment function of page level CTR approach ($\beta = 16$ since the AES block size is 16 bytes). After finding the necessary counter value, the CTR4 is used to decrypt the sensitive attribute of record 1. Then, the same operation is repeated for record 2. After reading the counter value from the beginning of the page, it is incremented by δ_2 where $\delta_2 = \lfloor \frac{\alpha_2}{\beta} \rfloor = \lfloor \frac{430}{16} \rfloor = 26$. Using this value, the sensitive attribute of record 2 is decrypted with CTR4.

Here, we illustrated decrypting the sensitive attributes of two records in a page. However, this operation will be repeated for all records in every page, if there is a projection query that needs to read all of the records in a table.

In the next section, the performance of mini page and page level CTR methods are compared based on our experiment results. In these experiments, we observed that selective decryption aspect of page level CTR causes significant performance gain under different query types.

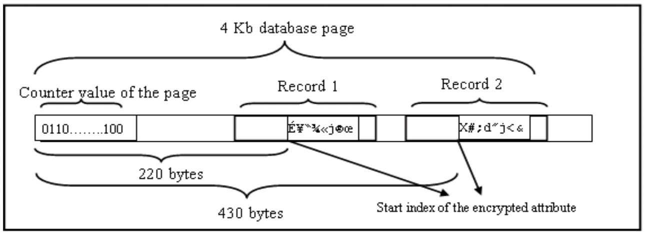


Fig. 3. Illustration of Page Level CTR approach

3.1 Experiments and Analyses

In order to compare the performance of Mini page and Page level CTR methods, we conducted some experiments. The first experiment is implemented to analyze the performance when all incoming queries to database are projection queries. The second experiment differs from experiment 1 since the queries are selection queries.

In both of these experiments it is assumed that the tuple lengths are 500 bytes and 70% of the tuples (350 byte/record) are encrypted. In each run, a 512 MB file is read sequentially from the disk and processed. In each read operation, 4 KB data is read from the disk since it is the default page size in many DBMSs.

The mini page method is implemented as it is discussed in the previous section. When a page is read from the disk, only the first part of the page is decrypted. Since all the sensitive attributes of the records are located at the beginning of the page, the decryption process continues until all sensitive attributes are decrypted.

On the other hand, page level CTR method is implemented by consecutively locating each record, which has sensitive and nonsensitive attributes. As it is discussed before, when a particular sensitive attribute of a record needs to be read, the counter value of that page is incremented as much as needed. Then CTR4 is used to decrypt the required sensitive attribute.

Projection Experiments: To observe the selective decryption property of page level CTR encryption method, we wanted to analyze the performance of projection queries in this experiment. As it is shown in figure 4, in the page level CTR method, the time required to decrypt the encrypted projection attribute is proportional to the length of the attribute. However, it is independent in the mini page level since this method decrypts all sensitive attributes to read the projection attributes of tuples. In contrast, page level CTR, decrypts as much data as needed to be decrypted. Therefore, both of the techniques require almost the same amount of time when the projection attribute is the only sensitive attribute in the record.

As a result of this experiment, we observed that, compared to mini page level encryption, page level CTR has a significant impact on reducing the cost of projection queries.

Selection Experiments: In projection experiments of page level CTR we were decrypting a certain amount of data for each record. However, in this experiment, in addition to selection attribute, we may need to decrypt the rest of the other sensitive attributes if the selection criterion is satisfied during query processing. This can be illustrated via an example, in which we have a table T with attributes A1, A2, and A3, where A1 and A2 are encrypted but A3 is not. In order to process a query such as "SELECT * FROM T WHERE A1 = X", the attribute A1 of each tuple should be decrypted. If the result of the condition A1 = X is true, not only A1 but also A2 should be decrypted. So, the performance of page level CTR depends on the selection condition of each tuples. Because of this reason, we repeated the experiments for different probability values of selecting a tuple.

The results of the experiments are shown in Figure 5, 6, and 7 where the probability of selecting a tuple is 30%, 60% and 100% respectively.

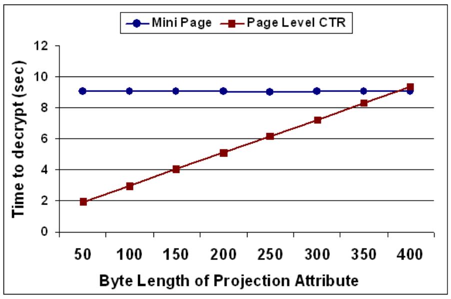


Fig. 4. Experiment results for projection queries

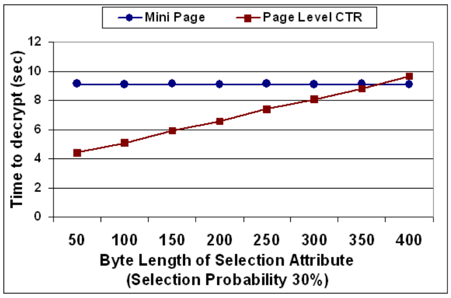


Fig. 5. Experiment results for selection queries

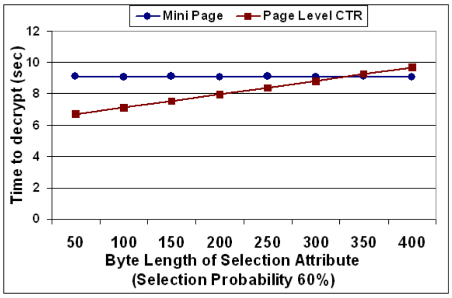


Fig. 6. Experiment results for selection queries

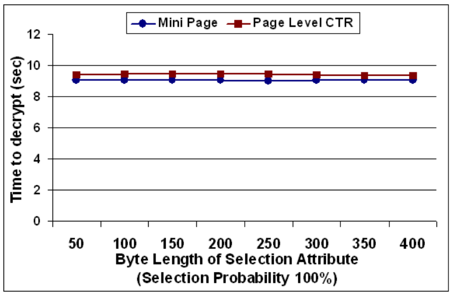


Fig. 7. Experiment results for selection queries

An important point to note is that the time required to decrypt tuples increases gradually when the probability of selecting tuples increases. When the probability is 100%, the cost of decrypting tuples becomes independent of the byte length of selection attribute. In addition, the cost of decrypting tuples in page level CTR is slightly more than the cost associated with the mini page as it is seen in Figure 7. The main reason for this is the overhead of extra index calculations in page level CTR approach.

In the selection experiment, we observed that selective decryption feature of page level CTR causes significant performance gain if the selection probability is low.

4 Discussion

We now discuss other encrypted data related issues in database management systems.

- **Key Management:** Using careful implementation of CTR mode, we can encrypt up to 2^{128} bits of data with a single key. Therefore, for most cases, we only need one encryption key per table. These encryption keys must be either stored in tamper-proof hardware or encrypted with a master key. If the master key option is chosen, during the system start, this master key can be loaded by the database administrator. If we do not want to trust the DB administrator with the master key, we can use classic threshold schemes to store the master key. For example, using a (k, t) secret sharing scheme, we can distribute this master key to t people and any k or more of them can come together to construct the master key [24].

  For security purposes, in the CTR mode, the same counter value should not be used for encrypting two different blocks. A simple way to solve this problem is to maintain a global counter value for each encryption key in use and update the counter value after each incrementation.
- **Insertion, Deletion, and Updates:** As mentioned above, counter values in CTR mode could not be used again. At the same time, we keep one initial counter value per page and calculate the counter values needed for selectively decrypting some attributes of the tuples using this initial counter values. When we need to update a part of the page, we need to encrypt the entire page with a different initial counter value.
- **Transaction Management** When there is an insert, delete, or update operation, DBMS will write a log to the log file. Therefore, to protect the sensitive data, we also need to encrypt the log file pages corresponding to encrypted tables. Otherwise, sensitive data values could be extracted from log files.

5 Conclusions

In this paper, we discussed the performance of different block cipher modes, under different encryption granularity and disk access patterns. Based on our experiments and analyses, we suggested a CTR based approach for encrypting data in DBMSs. We showed its potential for processing encrypted data faster by starting decryption process even before seeing the encrypted data. In addition to that, we proposed a page level encryption method by utilizing the selective decryption feature of CTR mode. Based on the results of our experiments, we compared the performance of the encryption method that we propose with the performances of other approaches and show its advantages under different query types. As a future work, we plan to implement our proposed approach using cryptographic accelerators in a distributed database environment.

Acknowledgments

We wish to thank Chris Clifton, Rakesh Agrawal and Sharad Mehrotra for helpful discussions.

References

1. Jr, T.Z.: An ominous milestone: 100 million data leaks. New York Times (December 18,2006)
2. Trinanes, J.A.: Database security in high risk environments.Technical report, governmentsecurity.org (2005) http://www.governmentsecurity.org/articles/DatabaseSecurityinHighRiskEn%vironments.php.
3. Standard for privacy of individually identifiable health information. Federal Register 67(157), 53181–53273 (2002)
4. California database security breach notification act (September 2002), http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_%_20020926_chaptered.html
5. Microsoft: Security features in microsoft sql server 2005. Technical report, Microsoft Corporation (2005), http://www.microsoft.com/sql/2005/productinfo/
6. IBM: Ibm data encryption for ims and db2 databases. Technical report, IBM Corporation (2006), http://www-306.ibm.com/software/data/db2imstools/db2tools/ibmencrypt.html
7. Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order-preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004, ACM Press, New York (2004)
8. Bayer, R., Metzger, J.K.: On the encipherment of search trees and random access files. ACM Trans. Database Syst. 1(1), 37–52 (1976), http://doi.acm.org/10.1145/320434.320445
9. Hardjono, T., Seberry, J.: Search key substitution in the encipherment of b-trees. In: McLeod, D., Sacks-Davis, R., Schek, H.J. (eds.) 16th International Conference on Very Large Data Bases, Brisbane, Queensland, Australia, Proceedings, August 13-16, 1990, pp. 50–58. Morgan Kaufmann (1990)
10. Hacigumus, H., Iyer, B.R., Li, C., Mehrotra, S.: Executing SQL over encrypted data in the database-service-provider model. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 4-6, 2002, pp. 216–227. ACM Press, New York (2002), http://doi.acm.org/10.1145/564691.564717

11. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: Proceedings of the 30th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., San Francisco (2004)
12. Damiani, E., Vimercati, S.D.C., jodia, S.J., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational dbmss. In: Proceedings of the 10th ACM conference on Computer and communications security, pp. 93–102. ACM Press,New York (2003), http://doi.acm.org/10.1145/948109.948124
13. Iyer, B., Mehrotra, S., Mykletun, E., Tsudik, G., Wu, Y.: A framework for efficient storage security in rdbms. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., Ferrari, E. (eds.) EDBT 2004. LNCS, vol. 2992, Springer, Heidelberg (2004)
14. Elovici, Y., Shmueli, E., nberg, R.W., Gudes, E.: A structure preserving database encryption scheme. In: Jonker, W., Petković, M. (eds.) SDM 2004. LNCS, vol. 3178, Springer, Heidelberg (2004), http://www.extra.research.philips.com/sdm-workshop/RonenSDM.pdf
15. NIST: Advanced encryption standard (aes). Technical Report NIST Special Publication FIPS-197, National Institute of Standards and Technology (2001), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
16. Recommendation for block cipher modes of operation methods and techniques. Technical Report NIST Special Publication 800-38A, National Institute of Standards and Technology (2001), http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf
17. Data encryption standard (des). Technical Report FIPS PUB 46-2, National Institutes of Standards and Technology (1988)
18. Schneier, B.: The blowfish encryption algorithm. Dr. Dobb's Journal, 38–40 (April 1994)
19. Lipmaa, H., Rogaway, P., Wagner, D.: Ctr-mode encryption. In: NIST, Computer Security Resource Center, First Modes of Operation Workshop (2000), http://csrc.nist.gov/CryptoToolkit/modes/workshop1/papers/lipmaa-ctr.pdf
20. Cox, M., Engelschall, R., Henson, S., Laurie, B.: The OpenSSL Project, http://www.openssl.org/
21. IBM: Table Space Design, http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com%.ibm.db2.udb.admin.doc/doc/c0004935.htm
22. Lipmaa, H.: Idea: A cipher for muldimedia architectures? In: Tavares, S., Meijer, H. (eds.) Selected Areas in Cryptography '98, Springer-Verlag,Heidelberg (1998)
23. Ailamaki, A., DeWitt, D.J., Hill, M.D., Skounakis, M.: Weaving relations for cache performance. In: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 169–180. Morgan Kaufmann Publishers Inc.,San Francisco (2001)
24. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)

Dynamic Event-Based Access Control as Term Rewriting[⋆]

Clara Bertolissi[2], Maribel Fernández[1], and Steve Barker[1]

[1] King's College London, Dept. of Computer Science, London WC2R 2LS, U.K.
[2] LIF, Université de Provence, Marseille, France
{Steve.Barker,Clara.Bertolissi,Maribel.Fernandez}@kcl.ac.uk

Abstract. Despite the widespread adoption of Role-based Access Control (RBAC) models, new access control models are required for new applications for which RBAC may not be especially well suited and for which implementations of RBAC do not enable properties of access control policies to be adequately defined and proven. To address these issues, we propose a form of access control model that is based upon the key notion of an event. The access control model that we propose is intended to permit the representation of access control requirements in a distributed and changing computing environment, the proving of properties of access control policies defined in terms of our model, and direct implementations for access control checking.

1 Introduction

Included amongst the most important problems in access control are the problems of formally defining richly expressive access control models that enable security administrators to specify a wide range of policies, using declarative languages, to prove properties of access control policies (for assurance purposes), and to evaluate access requests efficiently with respect to a representation of an access control policy. The increased use of access control policies in distributed computing environments has increased the need to have formal access control policies that are declarative (to handle the complexities of policy specification), to prove properties of policies (for verifiability purposes), and for efficient evaluation (given the computational overheads of potentially accessing large volumes of data from multiple locations). Moreover, given the complexities and scope involved, distributed applications have increased the requirements for autonomous changing of access control policies.

In recent years, work on RBAC [28,8] has emerged as *the* principal type of access control model in theory and in practice. RBAC is well suited for use with relatively static, closed, centralized systems where the assignment of a known, identifiable user to a role is specified by (typically) a centrally organized team of

[⋆] Research partially funded by the EU project *Implementing access control mechanisms using rewriting techniques*, Marie Curie Intra European Fellowships Programme.

human security administrators that has complete information about user qualifications and responsibilities, and hence user assignments to well-defined job functions and thus roles. Administrators are also (typically) assumed to have complete information about the permissions to be assigned to roles. The administrators of an RBAC policy for a centralized system will revise a policy formulation to take into account any changes to role and permission assignments (as required to meet organizational demands). These changes do not usually need to be performed in real-time; in general, RBAC policy specifications for centralized systems are relatively static (i.e., user-role, permission-role, and role-role relationships often persist for long periods of time). The features of RBAC have been selected to map onto organizational structures, and RBAC policies, in the centralized case, are mandatory in the sense that a user's access privileges on information sources is determined by the job function the user performs within the organization.

The features of RBAC that make it suitable for use in the centralized case are not necessarily so relevant in certain distributed computing contexts. In certain distributed environments, entities that request access to resources may not be known to enterprises with resources to protect (i.e., users may be stranger agents), *open* access policies are natural to adopt [8], as well as closed policies, decisions on access are more likely to need to be delegated to third-parties (e.g., in the case where the information about the identity or attributes of requesters may be required), the qualifications and responsibilities of requesters do not necessarily have any significance, in terms of access control, and the notion of a job function may not apply (as requesters for access to an enterprise's resources may have no connection with the enterprise). Although the features of RBAC naturally map to organizational structures, for many distributed applications the concept of an organizational structure may be irrelevant. The size, complexity and dynamic nature of some distributed systems present particular challenges that demand that changes to access policies be made frequently (e.g., in response to sales patterns and volumes) and by autonomous means (rather than by human security administrators manually modifying policy specifications). In the decentralized case, modifications to access policies for protecting an organization's assets may need to be made in response to events in external environments over which a policy administrator has no control, and about which administrators may not have complete information. Moreover, the high complexity of access control policies in the decentralized case demands not only that rich forms of language be used to represent these requirements but also that effective proof methods be employed to guarantee satisfaction of properties of policies.

To address the requirements for formal access policy representation for dynamic, distributed information systems, we propose an event-based distributed access control model, we demonstrate how and why access control policies, defined in terms of our model, should be considered as *term rewrite systems* [16,23,4], and we introduce *distributed term rewriting systems*. The model that we propose, and its representation using term rewriting, contributes to the literature on formal access control models by demonstrating how access control models may be defined that enable the autonomous changing of access control policies, the proving of properties of policies, and the efficient evaluation of access requests when the sources of access control and user requested information may be widely dispersed. We call the access control model that we introduce the *Dynamic Event-Based Access Control (*DEBAC*)* model. The *DEBAC* model addresses a number of limitations of RBAC when the latter is applied in distributed computational contexts: that in certain distributed environments, entities that request access to resources may not be known (so it is not possible to authorize access on the basis of a job function/role); that user authorizations may change dynamically on the basis of the occurrence of a wider range of events than the role and permission assignments used in RBAC; and that the information that is used to decide on granting/denying a user's request may be distributed across several sites (rather than being centrally located). We also demonstrate that the expressiveness of the *DEBAC* model and the representation of *DEBAC* policies, as term rewrite systems, permit a range of properties to be proven of *DEBAC* policies; these proofs guarantee that security goals are satisfied by a policy specification and by the operational methods used to evaluate access requests. Approaches that provide for provably correct security have always been and remain of high interest in the security community.

Represented as term rewrite systems, *DEBAC* policies are specified as a set of rules and access requests are specified as terms. Access request evaluation is effected by "reducing" terms to a *normal form* (see below). Term rewriting techniques have been successfully applied, and have had deep influence, in the development of computational models, programming and specification languages, theorem provers, and proof assistants. More recently, rewriting techniques have also been fruitfully exploited in the context of security protocols (see, for instance, [9]), and security policies for controlling information leakage (see, for example, [18]). As we will see, representing *DEBAC* policies as term rewriting systems enables complex and changing access control requirements to be succinctly specified in a declarative language that is formally well defined. The formal foundations on which our approach is based make it possible to apply the extensive theory of rewriting to access control; in particular, standard rewriting techniques can be used to show that access control policies satisfy essential properties (such as consistency and completeness) and can be used to study combinations of policy specifications. Another important reason to use rewrite-based languages to specify access control policies is that tools, such as ELAN [14,22] and MAUDE [15], can be used to test, compare and experiment with access request evaluation strategies, to automate equational reasoning, and for the rapid prototyping of access control policies.

The rest of this paper is organized as follows. In Section 2, we give some details on term rewriting to help to make the paper self-contained. In Section 3, we describe the *DEBAC* model and we introduce distributed rewrite systems as a tool to define *DEBAC* policies. In Section 4 we show how *DEBAC* policies can be specified via rewrite rules and we use this specification for proving essential properties of these policies. Some extensions of *DEBAC* policies are investigated in Section 5. We discuss related work in Section 6. Finally, we draw conclusions and make suggestions for further work in Section 7.

2 Preliminaries

In this section we recall some basic notions and notations for term rewriting. We refer the reader to [4] for additional information.

Term rewriting systems can be seen as programming or specification languages, or as formulae manipulating systems. They have been used in various applications (e.g., operational semantics, program optimization, automated theorem proving, and recently, computer security). We recall briefly the definition of first-order terms and term rewriting systems.

A *signature* $\mathcal{F}$ is a finite set of *function symbols* together with their (fixed) arity. $\mathcal{X}$ denotes a denumerable set of *variables* $X_1, X_2, \ldots$, and $T(\mathcal{F}, \mathcal{X})$ denotes the set of *terms* built up from $\mathcal{F}$ and $\mathcal{X}$.

Terms are identified with finite labeled trees. The symbol at the root of t is denoted by $root(t)$. *Positions* are strings of positive integers. The *subterm* of t at position p is denoted by $t|_p$ and the result of replacing $t|_p$ with u at position p in t is denoted by $t[u]_p$.

$\mathcal{V}(t)$ denotes the set of variables occurring in t. A term is *linear* if variables in $\mathcal{V}(t)$ occur at most once in t. A term is *ground* if $\mathcal{V}(t) = \emptyset$. Substitutions are written as in $\{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\}$ where t_i is assumed to be different from the variable X_i. We use Greek letters for substitutions and postfix notation for their application. We say that two terms unify if there is some substitution that makes them equal. Such a substitution is called a *unifier*. The *most general unifier* (mgu) is the unifier that will yield instances in the most general form.

Definition 1. *Given a signature $\mathcal{F}$, a* term rewriting system *on $\mathcal{F}$ is a set of rewrite rules $R = \{l_i \to r_i\}_{i \in I}$, where $l_i, r_i \in T(\mathcal{F}, \mathcal{X})$, $l_i \notin \mathcal{X}$, and $\mathcal{V}(r_i) \subseteq \mathcal{V}(l_i)$. A term t* rewrites *to a term u at position p with the rule $l \to r$ and the substitution σ, written $t \to_p^{l \to r} u$, or simply $t \to_R u$, if $t|_p = l\sigma$ and $u = t[r\sigma]_p$. Such a term t is called* reducible*. Irreducible terms are said to be in* normal form.

We denote by $\to_R^+$ (resp. $\to_R^*$) the transitive (resp. transitive and reflexive) closure of the rewrite relation $\to_R$. The subindex R will be omitted when it is clear from the context.

Example 1. Consider a signature for lists of natural numbers, with function symbols:

- z *(with arity* 0*) and* s *(with arity* 1*, denoting the successor function) to build numbers;*
- nil *(with arity 0, to denote an empty list),* cons *(with arity 2, to construct non-empty lists),* head *and* tl *(with arity 1, to obtain the head and tail of a list, resp.), and* length *(also with arity 1, to compute the length of a list).*

The list containing the numbers 0 *and* 1 *is written:* $\mathsf{cons}(\mathsf{z}, \mathsf{cons}(\mathsf{s}(\mathsf{z}), \mathsf{nil}))$*, or simply* $[\mathsf{z}, \mathsf{s}(\mathsf{z})]$ *for short. We can specify the functions* head*,* tl *and* length *with rewrite rules as follows:*

$$\begin{aligned} \mathsf{head}(\mathsf{cons}(X, L)) &\to X \\ \mathsf{tl}(\mathsf{cons}(X, L)) &\to L \\ \mathsf{length}(\mathsf{nil}) &\to \mathsf{z} \\ \mathsf{length}(\mathsf{cons}(X, L)) &\to \mathsf{s}(\mathsf{length}(L)) \end{aligned}$$

Then we have a reduction sequence:

$$\begin{aligned} &\mathsf{length}(\mathsf{cons}(\mathsf{z}, \mathsf{cons}(\mathsf{s}(\mathsf{z}), \mathsf{nil}))) \to \mathsf{s}(\mathsf{length}(\mathsf{cons}(\mathsf{s}(\mathsf{z}), \mathsf{nil})) \to \\ &\mathsf{s}(\mathsf{s}(\mathsf{length}(\mathsf{nil}))) \to \mathsf{s}(\mathsf{s}(\mathsf{z})) \end{aligned}$$

Let $l \to r$ and $s \to t$ be two rewrite rules (we assume that the variables of $s \to t$ were renamed so that there is no common variable with $l \to r$), p the position of a non-variable subterm of s, and μ a most general unifier of $s|_p$ and l. Then $(t\mu, s\mu[r\mu]_p)$ is a *critical pair* formed from those rules. Note that $s \to t$ may be a renamed version of $l \to r$. In this case a superposition at the root position is not considered a critical pair.

A term rewriting system R is:

– *confluent* if for all terms t, u, v: $t \to^* u$ and $t \to^* v$ implies $u \to^* s$ and $v \to^* s$, for some s;
– *terminating* (or *strongly normalizing*) if all reduction sequences are finite;
– *left-linear* if all left-hand sides of rules in R are linear;
– *non-overlapping* if there are no critical pairs;
– *orthogonal* if R is left-linear and non-overlapping;
– *non-duplicating* if for all $l \to r \in R$ and $X \in \mathcal{V}(l)$, the number of occurrences of X in r is less than or equal to the number of occurrences of X in l.

For example, the rewrite system in Example 1 is confluent, terminating, left-linear and non-overlapping (therefore orthogonal), and non-duplicating.

A *hierarchical union* of rewrite systems consists of a set of rules defining some basic functions (this is called the *basis* of the hierarchy) and a series of *enrichments*. Each enrichment defines a new function or functions, using the ones previously defined. Constructors may be shared between the basis and the enrichments.

We recall a modularity result for termination of hierarchical unions from [19] (Theorem 14), which will be useful later:

> *If in a hierarchical union the basis is non-duplicating and terminating, and each enrichment satisfies a general scheme of recursion, where each recursive call in the right-hand side of a rule uses subterms of the left-hand side, then the hierarchical union is terminating.*

3 The *DEBAC* Model

In this section, we describe the principal components of the *DEBAC* model and introduce distributed term rewriting systems.

3.1 Features of *DEBAC* Models

We begin by defining some of the key sets of constants in the signature that we use in the formulation of the *DEBAC* model and *DEBAC* policies. Specifically, we require:

- A countable set $\mathcal{R}$ of *resources*, written $r_1, r_2, \ldots$.
- A countable set $\mathcal{A}$ of named *actions* $a_1, a_2, \ldots$.
- A countable set $\mathcal{U}$ of *user identifiers*, written $u_1, u_2, \ldots$.
- A countable set $\mathcal{C}$ of *categories* $c_0, c_1, \ldots$.
- A countable set $\mathcal{E}$ of *event identifiers* $e_1, e_2, \ldots$.
- A countable set $\mathcal{S}$ of *site identifiers*, we use Greek letters $\mu, \nu, \ldots$ as site identifiers.
- A countable set $\mathcal{T}$ of *time points.*

The fundamental notion on which our *DEBAC* model is based is that of an event. In the *DEBAC* model, events are happenings at an instance of time that involve users performing actions. We view events as structured and described via a sequence l of ground terms of the form $\mathsf{event}(e_i, u, a, t)$ where event is a data constructor of arity four, e_i ($i \in \mathbb{N}$) are constants in $\mathcal{E}$ (denoting unique event identifiers), $u \in \mathcal{U}$ identifies a user, a is an action associated to the event, and t is the time when the event happened. In the discussion that follows, we represent times as natural numbers in *YYYYMMDD* format, and we assume that time is bidirectional so that proactive and postactive changes may be made to represent access policy requirements, and past, present and future times can be used in our model to make access control decisions.

In the *DEBAC* model, users may request to perform actions on resources that are potentially accessible from any site in a distributed system. A user is assigned to a particular category (e.g., *normal users, preferred users*, etc) on the basis of the history of events that relate to the user. Access to resources is then defined in the following way:

> *A user $u \in \mathcal{U}$ is permitted to perform an action $a \in \mathcal{A}$ on a resource $r \in \mathcal{R}$ that is located at site $s \in \mathcal{S}$ if and only if u is assigned to a category $c \in \mathcal{C}$ to which a access on r has been assigned.*

In the *DEBAC* model, assignments of a user u to a category c are based on the occurrence of events that are recorded in the history of events relating to u. As we will see later, hierarchies of categories of users may also be naturally accommodated in the *DEBAC* model.

We formally specify the notion of permitted access in term rewriting form below. In a *DEBAC* specification, there are two kinds of functions, which we call *generic* and *specific*, respectively. Generic functions are common to all *DEBAC* specifications, whereas specific functions, as their name suggests, depend on the specific scenario that we are modeling.

Given an event e_i, we will use standard generic functions to extract the component information from an event description. For instance, we may define a function user that returns the user involved in a given event, as follows:

$$\mathsf{user}(\mathsf{event}(E, U, A, T)) \to U$$

We assume that events are atomic, however, our model could be generalized to permit the representation of events that take place over a period of time and events that are composed of atomic parts (sub-events).

Also, we have chosen to include the time as an explicit component of an event. In certain contexts, it is sufficient to know the order of events. In such cases, the position of the event in a list of events provides enough information and the time parameter may be omitted.

3.2 Distributed Term Rewriting Systems

An important aspect of the *DEBAC* model is the capability of representing systems where resources may be distributed across different sites, and the information needed to decide whether a user request is granted or denied may also be distributed. To address this issue, we will define access control policies as modular term rewriting systems, where modules may be independently maintained at different sites, and information sources may be explicitly specified. In other words, policy designers may directly define the sites (locations) to be used in access request evaluation.

For the approach to distributed access control that we propose, we introduce distributed term rewriting systems (DTRSs); DTRSs are term rewriting systems where rules are partitioned into modules, each associated with a site, and function symbols are annotated with site identifiers. We assume that each site has a unique identifier (we use Greek letters $\mu, \nu, \ldots$ to denote site identifiers).

We say that a rule $f(t_1, \ldots, t_n) \to r$ defines f. There may be several rules defining f; we will assume that they are all at the same site ν. We write f_ν to indicate that the definition of the function symbol f is stored in the site ν. If a symbol is used in a rule without a site annotation, we assume the function is defined locally.

For example, in a DTRS used in a bank scenario, we may have a local function $\mathsf{account}$ such that $\mathsf{account}(u)$ returns u's bank account number, and rules computing the average balance of a user's account, stored in a site ν. Then we could define the security category of a user u using a rule

$$\begin{aligned}\mathsf{category}(U) \to\ & \mathit{if}\ \mathsf{averagebalance}_\nu(\mathsf{account}(U)) \geq 10000 \\ & \mathit{then}\ \mathsf{VIP\ CLIENT} \\ & \mathit{else}\ \mathsf{NORMAL\ CLIENT}\end{aligned}$$

The example above describes one instance of rule specification to illustrate the use of annotations on function symbols (we redefine categories for users in a more general context in the next section). Here, to calculate u's security category as a client, the average balance of u's account has to be computed at site ν, but u's account number is available locally. We use the notation $\mathit{if}\ \mathsf{b}\ \mathit{then}\ \mathsf{s}\ \mathit{else}\ \mathsf{t}$ as syntactic sugar for the term if-then-else(b, s, t), with the rewrite rules:

$$\begin{aligned}\text{if-then-else}(\mathsf{true}, X, Y) &\to X \\ \text{if-then-else}(\mathsf{false}, X, Y) &\to Y\end{aligned}$$

The syntax used in this paper to associate sites to function definitions is just one of many alternative notations (another alternative is to use an object-oriented inspired syntax, writing $\nu.\mathsf{averagebalance}(u)$).

In this paper, we assume that the site where each function is defined is known and therefore the annotations used in function symbols are just constants. An interesting generalization consists of allowing the use of variables as annotations when the site is not known in advance, and considering the dynamic computation of site identifiers (for instance, a 'linker' program could dynamically generate the address of the site where averagebalance is defined).

4 *DEBAC* Policy Specifications Via Rewrite Rules

In this section, we use an example to illustrate the use of distributed rewriting systems for specifying *DEBAC* policies. We do not claim that this is the only way to formalize a *DEBAC* policy as a rewrite system. Instead, our goal is to give an executable[1] specification of a *DEBAC* policy, to show some basic properties, and to address, using rewriting techniques, the problem of checking that the specification is consistent, correct, and complete (that is, no access can be both granted and denied, no unauthorized access is granted and no authorized access is denied).

4.1 Defining *DEBAC* Policies

We assume that events are represented as ground terms of the form $\mathsf{event}(e_i, u, a, t)$, as discussed above. We specify a *DEBAC* policy by giving its *generic* and *specific* functions.

The generic functions are category and status, together with auxiliary functions such as user (see Section 3). We define these functions by the rules:

$$\begin{aligned}
\mathsf{category}(U, L) &\to F(\mathsf{status}(U, L))\\
\mathsf{status}(U, \mathsf{nil}) &\to \mathsf{cons}(c_0, \mathsf{nil})\\
\mathsf{status}(U, \mathsf{cons}(E, L)) &\to \mathit{if}\ U = \mathsf{user}(E)\\
&\ \mathit{then}\ \mathsf{cons}(\mathsf{Estatus}(E)), \mathsf{status}(U, L))\\
&\ \mathit{else}\ \mathsf{status}(U, L)
\end{aligned}$$

where c_0 is a default category (we could return the empty list instead), status looks for events involving a user U in the list L, and uses a *specific* function Estatus that associates a category c_i to a user according to the particular event e_i in which the user was involved (this, of course, is specific to the application that we are modeling). The auxiliary function user extracts the user involved in a given event, as explained above. The function F, which is used in the definition of category, takes as argument a list of categories associated to a user, according to the history of events, and returns one specific category. Again, the particular

[1] For instance, the language MAUDE [15] can be used to execute rewrite-based specifications.

definition of F depends on the application. For example, we can take F to be the function head that returns the head of the list, or the functions max or min returning the highest or lowest category in the list, respectively; more elaborate functions are also possible.

Although we are focusing on first-order rewriting in this paper, the discussion above highlights an advantage of a higher-order rewriting formalism (such as, e.g., Combinatory Reduction Systems [24]). Indeed, in a higher-order rewriting system, category and status would be parameterized by F and Estatus, respectively (i.e., they would be variables that can be instantiated with different functions).

Specific functions, as their name suggests, depend on the specific application that we are modeling. For example, in a bank scenario, we may define:

$$\begin{array}{l}\mathsf{Estatus}(\mathsf{event}(E, U, depositing, T)) \\ \rightarrow if\ \mathsf{averagebalance}_{\nu}(\mathsf{account}(U)) > 10000\ and\ \mathsf{NotBlacklisted}_{\mu}(U) \\ then\ \mathsf{GOLD\text{-}CLIENT}\ else\ \mathsf{NORMAL\text{-}CLIENT}\end{array}$$

whereas in a university, students may acquire rights (change category) as they pass their exams:

$$\begin{array}{ll}\mathsf{Estatus}(\mathsf{event}(E, U, enroll, T)) & \rightarrow \text{REGISTERED-STUDENT} \\ \mathsf{Estatus}(\mathsf{event}(E, U, pay, T)) & \rightarrow \text{REGULAR} \\ \mathsf{Estatus}(\mathsf{event}(E, U, exams1styear, T)) & \rightarrow if\ \mathsf{pass}_{\nu}(U, 1styear) \\ & \quad and\ \mathsf{paid}_{\mu}(U, fees) \\ & \quad then\ \text{2ND-YEAR STUDENT} \\ & \quad else\ \text{IRREGULAR}\end{array}$$

where pass_{ν} and paid_{μ} are auxiliary functions returning boolean values.

Consider now the (chronologically ordered) list of events

$$\begin{array}{l} l = [\ \mathsf{event}(e_2, u, exams1styear, 20060130), \mathsf{event}(e_1, u, pay, 20060115), \\ \quad\ \ \mathsf{event}(e_0, u, enroll, 20050901)]\end{array}$$

and assume that the function F, that is used in the definition of category, is the function head returning the head of a list. Then we have

$$\begin{array}{l}\mathsf{category}(u, l) \rightarrow \mathsf{head}(\mathsf{status}(u, l)) \rightarrow^* \\ \mathsf{head}([\text{2ND-YEAR STUDENT, REGULAR, REGISTERED-STUDENT}])\end{array}$$

which finally leads to $\mathsf{category}(u, l) \rightarrow^*$ 2ND-YEAR STUDENT.

4.2 Evaluating Access Requests

Access requests from users can be evaluated by using a rewrite system to grant or deny the request according to the history of events and the user's category assignments that are specified in the *DEBAC* policy. For that, we may use the following rules, where a user u asks for an action a to be performed on a resource r accessible from a site μ. The symbols U, A, R, S, L are variables and the operator member is the standard membership test operator.

$$\begin{aligned} \mathsf{access}(A, U, R, S, L) &\to \mathsf{check}(\mathsf{member}((A, \mathsf{category}(U, L)), \mathsf{privileges}(R, S))) \\ \mathsf{check}(\mathsf{true}) &\to \mathsf{grant} \\ \mathsf{check}(\mathsf{false}) &\to \mathsf{deny} \end{aligned}$$

Here we assume that the function privileges returns a list of pairs (action, category allowed to perform that action) for a given resource in a given site. For example, privileges may be defined by rules such as:

$$\mathsf{privileges}(r, s) \to [(a_{11}, c_{11}), \ldots, (a_{1n}, c_{1n})]$$

In the discussion that follows, we will use R_{DEBAC} to refer to the rewrite system that contains the set of rules that we have defined so far.

4.3 Properties of the *DEBAC* Policy

In order for a *DEBAC* policy to be "acceptable", it is necessary that the policy satisfies certain acceptability criteria. As an informal example, it may be necessary to ensure that an access policy formulation does not specify that any user is granted and denied the same access privilege on the same data item (i.e., that the policy is consistent).

The following properties of R_{DEBAC} are easy to check and will be used to show that the specification is consistent, correct and complete:

Property 1. The rewrite system R_{DEBAC} is terminating and confluent.

Proof. i) To prove termination, we use a modularity result for hierarchical unions (see Section 2 and [19]). First, observe that the system R_{DEBAC} is hierarchical: The auxiliary functions such as user, account, averagebalance, pass, paid, etc., form the basis of the hierarchy; they are terminating and non-duplicating. The specific rules defining Estatus, privileges and check form the first enrichment, and they are clearly terminating (they are not recursive). The second enrichment consists of the rules defining status and auxiliary functions, such as member. These are recursive functions, but the recursive calls are made on strict subterms of the arguments in the left-hand side of the rule. Finally, the non-recursive rules, defining category and access, complete the system.
We can therefore conclude that the system is terminating [19].

ii) To prove confluence, first note that there are no critical pairs, therefore the system is locally confluent. Termination and local confluence imply confluence, by Newman's Lemma [27].

Corollary 1. *Every term has a unique normal form in* R_{DEBAC}.

As a consequence of the unicity of normal forms, our specification of the *DEBAC* policy R_{DEBAC} is *consistent.*

Property 2 (Consistency). For any list of events l, $u \in \mathcal{U}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, $s \in \mathcal{S}$: it is not possible to derive, from R_{DEBAC}, both grant and deny for a request $\mathsf{access}(a, u, r, s, l)$.

We can give a characterization of the normal forms:

Property 3. The normal form of a ground term of the form $\mathsf{access}(a, u, r, s, l)$ where $u \in \mathcal{U}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, $s \in \mathcal{S}$ and l is a list of events, is either grant or deny.

As a consequence, our specification of the access control policy is *total*.

Property 4 (Totality). Each access request $\mathsf{access}(a, u, r, s, l)$ from a pre-authenticated user u to perform an action a on the resource r in a site s is either granted or denied.

Correctness and *Completeness* are also easy to check:

Property 5 (Correctness and Completeness). For any $u \in \mathcal{U}$, $a \in \mathcal{A}$, $r \in \mathcal{R}$, $s \in \mathcal{S}$ and list of events l:

- $\mathsf{access}(a, u, r, s, l) \rightarrow^* \mathsf{grant}$ if and only if u has the access privilege a on r in s.
- $\mathsf{access}(a, u, r, s, l) \rightarrow^* \mathsf{deny}$ if and only if u does not have the access privilege a on r in s.

Proof. Since the specification is consistent and total, it is sufficient to show that $\mathsf{access}(a, u, r, s, l) \rightarrow^* \mathsf{grant}$ if and only if u belongs to a category of users that is assigned the access privilege a on the resource r in s. This is easy to check in the examples above, by inspection of the rewrite rules.

It is important to note that the proofs above do not have to be generated by a security administrator; rather, the proofs demonstrate that a *DEBAC* policy R_{DEBAC} satisfies the properties described above. A security administrator can simply base a *DEBAC* policy on the term rewrite system that we have defined and can be sure that the properties of R_{DEBAC} hold.

The rewrite rules provide an executable specification of the policy (the rewrite rules are both a specification *and* an implementation of the access control function). The rules given above can be transformed into a MAUDE program by adding type declarations for the function symbols and variables used and by making minor syntactical changes.

5 Extensions of the *DEBAC* Model

5.1 *DEBAC* with Ordered Categories

In RBAC models, it is usual to include role hierarchies to allow for the implicit specification of authorizations. This idea can be incorporated into the *DEBAC* model: we can accommodate a notion of a hierarchy of categories for users, where a user in category c_i will inherit, via a category hierarchy, the privileges of users in any category c_j such that $c_j < c_i$ with respect to a given partial ordering.

To represent a partial ordering on categories, we can add rules of the form

$$\mathsf{dpred}(c_i) \to [c_1, \ldots, c_j]$$

to specify a function $\mathsf{dpred} : \mathcal{C} \to \mathit{List}(\mathcal{C})$, where $\mathsf{dpred}(c_i) = [c_1, \ldots, c_j]$ means that $c_1, \ldots, c_j$ are direct predecessors of c_i in the ordering. Then we redefine the member function used in the definition of access (see Section 4.2), so that it takes into account the ordering (i.e., a category will have its privileges plus the privileges of all the categories that precede it in the ordering).

We use the following definition of member (where we omit the definition of the standard operations on sets and booleans):

$$\begin{array}{rl}
\mathsf{member}((A, C), \mathsf{nil}) & \to \mathsf{false} \\
\mathsf{member}((A, C), \mathsf{cons}((A', C'), L)) & \to \mathit{if} \;\; A = A' \;\; \mathit{and} \;\; (C = C' \; \mathit{or} \; C' \in \mathsf{pred}(C)) \\
& \; \mathit{then} \;\; \mathsf{true} \\
& \; \mathit{else} \;\; \mathsf{member}((A, C), L)
\end{array}$$

$$\begin{array}{rl}
\mathsf{pred}(C) & \to \mathsf{dpred}(C) \cup \mathsf{preds}(\mathsf{dpred}(C)) \\
\mathsf{preds}(\mathsf{nil}) & \to \mathsf{nil} \\
\mathsf{preds}(\mathsf{cons}(C, L)) & \to \mathsf{pred}(C) \cup \mathsf{preds}(L)
\end{array}$$

Note that we do not need to change the definition of access to accommodate hierarchies of categories, and we do not need to impose conditions on the form that a hierarchy takes (apart from an acyclicity condition, which is a natural requirement for hierarchies).

5.2 *DEBAC* with Constraints

A number of RBAC models with constraints, such as *separation of duties*, have been proposed. Constraints that are similar to separation of duties constraints can also be specified in a *DEBAC* model using rewrite rules, as an administrative check on a *DEBAC* policy.

Separation of duties is the property that specifies that categories assigned to a user cannot be mutually exclusive. To ensure that a specification of a *DEBAC* policy satisfies the separation of duties property, we will erase conflicting categories assigned to a user (producing a list of non-mutual-exclusive categories). This is obtained by evaluation of $\mathsf{clean}(\mathsf{status}(u))$ in a rewrite system containing the rules:

$$\begin{array}{rll}
\mathsf{clean}(\mathsf{nil}) & \to \mathsf{nil} & \\
\mathsf{clean}(\mathsf{cons}(C, L)) & \to \mathsf{cons}(C, \mathsf{clean}(\mathsf{eraseclash}(C, L))) & \\
\mathsf{eraseclash}(C, \mathsf{nil}) & \to \mathsf{nil} & \\
\mathsf{eraseclash}(C, \mathsf{cons}(C', L)) & \to \mathsf{cons}(C', \mathsf{eraseclash}(C, L)) & (C, C' \text{ do not clash}) \\
\mathsf{eraseclash}(C, \mathsf{cons}(C', L)) & \to \mathsf{eraseclash}(L) & (C, C' \text{ clash})
\end{array}$$

6 Related Work

In this section, we discuss our approach in relation to existing related literature. We note first that reduction systems have been used to model a variety of problems in security. For example, the SPI-calculus [1] was developed as an extension of the π-calculus for proving the correctness of authentication protocols. The π-calculus itself has been used to reason about a number of basic access control policies and access mechanisms (see, for example, [3]). Term rewriting has also been used in the analysis of security protocols [9], for defining policies for controlling information leakage [18], and for intrusion detection [2].

On the use of rewriting for access control specifications, the work by Koch et al [25] is related to our proposal. In [25], Koch et al describe a formalization of RBAC using a graph-based approach, with graph transformation rules used for describing the effects of actions as they relate to RBAC notions. More recently, [7,29] use term rewrite rules to model access control policies. The formalization used by Koch et al provides a basis for proving properties of RBAC specifications, based on the categorical semantics of the graph transformations. The approach used in [7,29] is operational: access control policies are specified as sets of rewrite rules and access requests are specified as terms which are evaluated using the rewrite rules. Our work addresses similar issues to [7,25,29] but provides a different formulation of access policies that is suitable for distributed environments. We also define a new type of access control model, *DEBAC*, that we argue generalizes RBAC. On the latter point, RBAC may be viewed as a special case of *DEBAC* in which the only necessary events are those involving security administrators performing the actions of user role assignment and user role deassignment, and the actions of permission assignment (to a role) and permission deassignment (from a role). The events of consequence that involve users are events of activating a role and deactivating a role. These events can be naturally accommodated in the *DEBAC* model; however, the *DEBAC* model additionally permits any number of other events to be represented.

The work on access control by Jajodia et al [20] and Barker and Stuckey [8] is also related to ours. In [20] and [8], access control requirements are represented in (constraint) logic programming languages. In these approaches, the requirements that must be satisfied in order for requesters to access resources are specified by using rules expressed in (C)LP languages and access request evaluation may be viewed as being performed by reducing an access request to a non-reducible clause (using, for example, SLG-resolution [31] or constraint solvers [26]). The term rewriting approach is similarly based on the idea of computation by reduction, and has similar attractions to the (C)LP approaches of Jajodia et al and Barker and Stuckey. However, in contrast to these approaches, our proposal does not require that the syntactic restriction to access policies that are *locally stratified* [6] be adopted (to ensure the existence of a categorical semantics and thus unambiguous access control policies). Moreover, we describe a form of access control model, the *DEBAC* model, that is applicable in the context of distributed access whereas [20] and [8] formally define access control models for centralized systems. The *DEBAC* model that we have described is also more expressive than any of the *Datalog*-based languages that have been proposed for distributed access control (see [5,21,17,10]); these languages, being based on a monotonic semantics, are not especially well suited for representing dynamically *changing* distributed access request policies of the form that we have considered in this paper.

Work on temporal RBAC [8,11], is related to our work on *DEBAC* in the sense that TRBAC models are concerned with the important notion of change and events. However, in [8] and [11], the events of interest are restricted to simple time/clock events. In the *DEBAC* model, any number of application-specific events (e.g., enrolling, passing_exams, etc.) may be represented, in addition to clock events. In [12], event expressions are used to trigger the enabling and disabling of roles. However, the proposal in [12] is based on an RBAC model that is quite different to the *DEBAC* model. Moreover, to ensure that a categorical semantics exists, syntactic restrictions are imposed on the language described in [12], to treat conflicting (prioritized) event expressions; such restrictions do not need to be imposed in our approach.

We also note that although our approach is based on rules and events the framework that we describe has a well defined declarative semantics that is quite different to the *ad hoc* operational semantics that are used in, for example, *active rule systems* [13]. Active rule systems are also based on the notions of rules and events but, unless some generally quite restrictive syntactic constraints are imposed, do not provide an adequate semantic basis for proving properties of access control policies (unlike term rewrite systems).

7 Conclusions and Further Work

We have described an event-based distributed access control model that we have developed to address certain shortcomings of RBAC models when the latter are applied in certain distributed computing contexts. Our *DEBAC* model takes the notion of an event as primitive (rather than a role), and has been designed to include features that specifically facilitate the autonomous changing of access control policy requirements, the proving of a wide range of properties of *DEBAC* policies (which follow directly from the syntax of *DEBAC* policy specifications), and the use of correct operational methods for the evaluation of user access requests with respect to distributed sources of access control and other forms of information (that do not need to be syntactically restricted in the way that other access policy specification languages are). For distributed access control, we introduced distributed term rewriting systems.

We note that the idea of access policy composition [30] is especially important in distributed environments (where access control information may need to be shared across multiple sites). Hence, a key matter for future work is to define appropriate algebras for *DEBAC* policy composition. A related matter for future work is to relax our assumption of atomic events and to treat access request evaluation in terms of sequences, disjunctions and conjunctions of events (for which an event algebra may be defined). We also intend to investigate the issue of evaluating access requests when conflicting information is received about the same user from different sites in a distributed system.

Acknowledgements

Part of this work was carried out at LIF, while the second author was visiting the MoVe team, partially supported by the Université de Provence.

References

1. Abadi, M., Gordon, A.: A calculus for cryptographic protocols: The spi calculus. In: Proc. 4th ACM Conf. on Computer and Communication Security, pp. 36–47. ACM Press, New York (1997)
2. Abbes, T., Bouhoula, A., Rusinowitch, M.: Protocol analysis in intrusion detection using decision tree. In: Proc. ITCC'04, pp. 404–408 (2004)
3. Abendroth, J., Jensen, C.: A unified security mechanism for networked applications. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 351–357. Springer, Heidelberg (2004)
4. Baader, F., Nipkow, T.: Term rewriting and all that. Cambridge University Press, Great Britain (1998)
5. Bacon, J., Moody, K., Yao, W.: A model of OASIS RBAC and its support for active security. TISSEC 5(4), 492–540 (2002)
6. Baral, C., Gelfond, M.: Logic programming and knowledge representation. JLP 20, 73–148 (1994)
7. Barker, S., Fernández, M.: Term rewriting for access control. In: Damiani, E., Liu, P. (eds.) Data and Applications Security XX. LNCS, vol. 4127, Springer, Heidelberg (2006)
8. Barker, S., Stuckey, P.: Flexible access control policy specification with constraint logic programming. ACM Trans. on Information and System Security 6(4), 501–546 (2003)
9. Barthe, G., Dufay, G., Huisman, M., de Sousa, S.M.: Jakarta: a toolset to reason about the JavaCard platform. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, Springer, Heidelberg (2001)
10. Becker, M., Sewell, P.: Cassandra: Distributed access control policies with tunable expressiveness. In: POLICY 2004, pp. 159–168 (2004)
11. Bertino, E., Bettini, C., Ferrari, E., Samarati, P.: An access control model supporting periodicity constraints and temporal reasoning. ACM TODS 23(3), 231–285 (1998)
12. Bertino, E., Bonatti, P., Ferrari, E.: TRBAC: A temporal role-based access control model. In: Proc. 5th ACM Workshop on Role-Based Access Control, pp. 21–30. ACM Press, New York (2000)
13. Bertino, E., Catania, B., Zarri, G.: Intelligent Database Systems. Addison-Wesley, Reading (2001)
14. Borovansky, P., Kirchner, C., Kirchner, H., Moreau, P-E.: ELAN from a rewriting logic point of view. TCS 285, 155–185 (2002)
15. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: The Maude 2.0 system. In: Nieuwenhuis, R. (ed.) RTA 2003. LNCS, vol. 2706, pp. 76–87. Springer, Heidelberg (2003)

16. Dershowitz, N., Jouannaud, J.-P.: Rewrite Systems. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science: Formal Methods and Semantics, vol. B, North-Holland, Amsterdam (1989)
17. De Treville, J.: Binder, a logic-based security language. In: Proc. IEEE Symposium on Security and Privacy, pp. 105–113. IEEE Computer Society Press, Los Alamitos (2002)
18. Echahed, R., Prost, F.: Security policy in a declarative style. In: Proc. PPDP'05, ACM Press, New York (2005)
19. Fernández, M., Jouannaud, J.-P.: Modular termination of term rewriting systems revisited. In: Reggio, G., Astesiano, E., Tarlecki, A. (eds.) Recent Trends in Data Type Specification. LNCS, vol. 906, Springer, Heidelberg (1995)
20. Jajodia, S., Samarati, P., Sapino, M., Subrahmaninan, V.S.: Flexible support for multiple access control policies. ACM TODS 26(2), 214–260 (2001)
21. Jim, T.: SD3: A trust management system with certified evaluation. In: IEEE Symp. Security and Privacy, pp. 106–115. IEEE Computer Society Press, Los Alamitos (2001)
22. Kirchner, C., Kirchner, H., Vittek, M.: ELAN user manual. Nancy (France), Technical Report 95-R-342, CRIN (1995)
23. Klop, J.-W.: Term Rewriting Systems. In: Abramsky, S., Gabbay, D.M., Maibaum, T.S.E. (eds.) Handbook of Logic in Computer Science, vol. 2, Oxford University Press, Oxford (1992)
24. Klop, J.-W., van Oostrom, V., van Raamsdonk, F.: Combinatory reduction systems, introduction and survey. TCS 121, 279–308 (1993)
25. Koch, M., Mancini, L., Parisi-Presicce, F.: A graph based formalism for rbac. In: Proc. SACMAT'04, pp. 129–187 (2004)
26. Marriott, K., Stuckey, P.J.: Programming with Constraints: an Introduction. MIT Press, Cambridge (1998)
27. Newman, M.H.A.: On theories with a combinatorial definition of equivalence. Annals of Mathematics 43(2), 223–243 (1942)
28. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
29. de Oliveira, A.S.: Rewriting-based access control policies. In: Proc. of SECRET'06. ENTCS, Elsevier, Amsterdam (2007)
30. Wijesekera, D., Jajodia, S.: Policy algebras for access control the predicate case. In: ACM Conf. on Computer and Communications Security, pp. 171–180. ACM Press, New York (2002)
31. The XSB System Version 2.7.1, Programmer's Manual (2005)

A Spatio-temporal Role-Based Access Control Model

Indrakshi Ray and Manachai Toahchoodee

Department of Computer Science
Colorado State University
{iray,toahchoo}@cs.colostate.edu

Abstract. With the growing advancement of pervasive computing technologies, we are moving towards an era where spatio-temporal information will be necessary for access control. The use of such information can be used for enhancing the security of an application, and it can also be exploited to launch attacks. For critical applications, a formal model for spatio-temporal-based access control is needed that increases the security of the application and ensures that the location information cannot be exploited to cause harm. In this paper, we propose a spatio-temporal access control model, based on the Role-Based Access Control (RBAC) model, that is suitable for pervasive computing applications. We show the association of each component of RBAC with spatio-temporal information. We formalize the model by enumerating the constraints. This model can be used for applications where spatial and temporal information of a subject and an object must be taken into account before granting or denying access.

1 Introduction

With the increase in the growth of wireless networks and sensor and mobile devices, we are moving towards an era of pervasive computing. The growth of this technology will spawn applications such as, the Aware Home [5] and CMU's Aura [7], that will make life easier for people. Pervasive computing applications introduce new security issues that cannot be addressed by existing access control models and mechanisms. For instance, access to a computer should be automatically disabled when a user walks out of the room. Traditional models, such as Discretionary Access Control (DAC) or Role-Based Access Control (RBAC) do not take into account such environmental factors in determining whether access should be allowed or not. Consequently, new access control models and mechanisms are needed that use environmental factors, such as, time and location, while determining access.

The use of spatial and temporal information for access can be used for enhancing the security of other applications as well. For instance, a user should be able to fire a missile from specific high security locations only. Moreover, the missile can be fired only when it is in a certain location. For such critical applications, we can include additional checks, such as the verification of the location of the user and that of the missile, that must be satisfied before the user is granted access. With the reduction in the cost of Global Positioning System, this is indeed a viable option.

In this paper, we propose a formal spatio-temporal model that is suitable for commercial applications. Since RBAC is policy-neutral, simplifies access management, and used by commercial applications, we decide to base our work on it. We show how RBAC can be extended to incorporate the notion of time and location. We illustrate how each component of RBAC is related with time and location. In other words, we explain how time and location impact each component of RBAC. Finally, we show how this spatio-temporal information can be used to determine whether an user has access to a given object. The correct behavior of this model is formulated in terms of constraints that must be satisfied by any application using this model.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 briefly illustrates how time and location are represented in our model. Section 4 shows the relationship of each component of Core RBAC with location. Sections 5, 6 and 7 propose different types of hierarchies and separation of duty constraints that we can have in our model. Section 8 discusses a simple example using our model. Section 9 concludes the paper with some pointers to future directions.

2 Related Work

Recently, some research attempts have been done under the area of context aware access control. Yu et al. [15] proposed LTAM a location-temporal authorization model, which is based on a Discretionary Access Control (DAC) model. The goal of this paper is different than ours; it focuses on controlling access to the different locations. For example, access rules may have temporal constraints that can specify when a user can enter or leave a location or how many times a user can enter a location. However, it does not address the issue of where and when a subject can access a given object. Since this model is based on DAC, authorization management is a problem.

Role-based access control model [6] is used for addressing the access control needs of commercial organizations. In RBAC permissions are attached to roles and users must be assigned to roles to get the permissions. Permissions determine what operations can be carried out on resources under access control. A user must establish a session to activate a subset of roles to which the user is assigned. Each user can activate multiple sessions, however, each session is associated with only one user. The operations that a user can perform in a session depend on the roles activated in that session and the permissions associated with those roles. RBAC also supports role hierarchies. Role hierarchies define an inheritance relationship between roles. To prevent conflict of interests that arise in an organization, RBAC allows the specification of Static and Dynamic Separation of Duty constraints.

Several work exists that improve RBAC functionality. We discuss only those that are very closely related to ours. Some of these work focus on how RBAC can be extended to make it context aware. Sampemane et al. [12] present a new access control model for active spaces. Active space denotes the computing environment integrating physical spaces and embedded computing software and hardware entities. The active space allows interactive exchange of information between the user and the space. Environmental aspects are adopted into the access control model for active spaces, and the space roles are introduced into the implementation of the access control model based on RBAC. The model supports specification of MAC policies in which system administrator maintains the access matrix and DAC policies in which users create and update security policies for their devices.

Covington et al. [5] introduce environment roles in a generalized RBAC model (GR-BAC) to help control access control to private information and resources in ubiquitous computing applications. The environments roles differ from the subject roles in RBAC but do have similar properties including role activation, role hierarchy and separation of duty. In the access control framework enabled by environment roles, each element of permission assignment is associated with a set of environment roles, and environment roles are activated according to the changing conditions specified in environmental conditions; in this way, environmental properties like time and location are introduced to the access control framework. In a subsequent work [4], Covington et al. describes the Context-Aware Security Architecture (CASA) which is an implementation of the GR-BAC model. The access control is provided by the security services in the architecture. In CASA, polices are expressed as roles and managed by the security management service, authentication and authorization services are used to verify user credentials and determine access to the system resources. The environmental role activation services manage environmental role activation and deactivation according to the environment variables collected by the context management services.

Other extensions to RBAC include the Temporal Role-Based Access Control Model (TRBAC) proposed by Bertino et al. [1]. This work adds the time dimension to the RBAC model. The authors in this paper introduce the concept of role enabling and disabling. Temporal constraints determine when the roles can be enabled or disabled. A role can be activated only if it has been enabled. Joshi et al.[8] extend this work by proposing the Generalized Temporal Role Based Access Control Model (GTRBAC). In this work the authors introduce the concept of time-based role hierarchy and time-based separation of duty. These works do not discuss the impact of spatial information.

Researchers have also extended RBAC to incorporate spatial information. The most important work in this regard is the GEO-RBAC [2]. In this model, role activation is based on the location of the user. For instance, a user can acquire the role of teacher only when he is in the school. Outside the school, he can acquire the role of citizen. The model supports role hierarchies but does not deal with separation of duties. Another work incorporating spatial information is by Ray et al. [11]. Here again, the authors propose how each component of RBAC is influenced by location. The authors define their formal model using the Z specification language. Role hierarchy and separation of duties are not addressed in this paper. None of these work discuss the impact of time on location. Location-based access control has been addressed in other works not pertaining to RBAC [7,9,10].

To the best of our knowledge, the only work which propose incorporating both time and location in RBAC is by Chandran et al. [3]. The paper combines the main features of GTRBAC and GEO-RBAC. Here again, role is enabled by time constraints. The user can activate the role if the role is enabled and the user satisfies the location constraints associated with role activation. Our current work is closely related to this work. The similarity is that in both the models role activation occurs when temporal and spatial constraints are satisfied. However, there are a number of points where we differ. First, in Chandran's work, role assignment is not dependent on location or time. A number of motivating examples indicate that role assignment should be dependent on role and time. Consequently, we incorporate this feature in our model. Second, in Chandran's work, when a role can be activated all the permissions associated with the role can be invoked. This may not be true in real world. For instance, a system administrator's role can be activated from 9:00 a.m. to 9:00 p.m. everyday. However, he can perform backup only during 8:00 to 9:00 p.m. on Fridays. Chandran's model cannot express this situation. We associate a permission with additional location and temporal constraints that must be satisfied before a permission can be invoked. Third, Chandran's work does not discuss the impact of location and time on role hierarchy or separation of duty. We propose different types of time and location based hierarchy and separation of duty constraints in our model which will be useful for real-world applications.

3 Representing Location and Time

3.1 Representing Location

In order to perform location-based access control, we need to perform operations on location information and protect the location information. In this section, we formalize the concept of location [2,3] and propose the location comparison operators that are used in our model.

There are two types of locations: *physical* and *logical*. All users and objects are associated with locations that correspond to the physical world. These are referred to as the physical locations. A physical location is formally defined by a set of points in a three-dimensional geometric space.

Definition 1. [Physical Location] *A physical location $PLoc_i$ is a non-empty set of points $\{p_i, p_j, \ldots, p_n\}$ where a point p_k is represented by three co-ordinates.*

Physical locations are grouped into symbolic representations that will be used by applications. We refer to these symbolic representations as logical locations. Examples of logical locations are Fort Collins, Colorado etc.

Definition 2. [Logical Location] *A logical location is an abstract notion for one or more physical locations.*

We assume the existence of two translation functions, m and m', that convert from logical locations to physical locations and vice-versa.

Definition 3. [Mapping Functions m and m'] *m is a total function that converts a physical location into a logical one. m' is a total function that converts a logical location into a physical one. Let P be the set of all possible physical locations and L be the set of all logical locations. The following formalizes the functions.*

- $m : P \rightarrow L$.
- $m' : L \rightarrow P$
- *For any logical location Loc_i, $m(m'(Loc_i)) = Loc_i$.*
- *For any physical location $PLoc_j$, $m'(m(PLoc_j)) = PLoc_j$.*

Different kinds of relationships may exist between a pair of locations. We discuss one such relationship, known as *containment*, that will be used in this paper. Containment formalizes the idea whether one location is contained within another. Intuitively, a physical location $ploc_j$ is contained in another physical location $ploc_k$, if all points in $ploc_j$ also belong to $ploc_k$. This is formalized as follows.

Definition 4. [Containment Relation] *A physical location $ploc_j$ is said to be contained in another physical location $ploc_k$, denoted as, $ploc_j \subseteq ploc_k$, if the following condition holds: $\forall p_i \in ploc_j, p_i \in ploc_k$. The location $ploc_j$ is called the contained location and $ploc_k$ is referred to as the containing or the enclosing location. A logical location $lloc_m$ is contained in $lloc_n$, denoted as, $lloc_m \subseteq lloc_n$, if and only if the physical location corresponding to $lloc_m$ is contained within that of $lloc_m$, that is $m'(lloc_m) \subseteq m'(lloc_n)$.*

Note that, a physical location may be contained in a logical location or vice-versa. In such cases, we use the mapping functions to convert the logical locations into physical ones and then test whether one is contained within the other.

We assume the existence of a logical location called *universe* that contains all other locations.

In the rest of the paper, we do not discuss physical locations any more. The locations referred to are logical locations.

3.2 Representing Time

Our model uses two kinds of temporal information. We feel it is necessary to distinguish between these two kinds of information because they have very different semantics. The first is known as time instant and the other is time interval. Time can be represented as a set of discrete points on the time line.

Definition 5. [Time Instant] *A* time instant *is one discrete point on the time line.*

The exact granularity of a time instant will be application dependent. For instance, in some application a time instant may be measured at the nanosecond level and in another one it may be specified at the millisecond level.

Definition 6. [Time Interval] *A* time interval *is a set of time instances. When the time instances making up an interval are consecutive, we refer to the interval as a* continuous *one. Otherwise, the interval is said to be* non-continuous.

Example of a continuous interval is 9:00 a.m. to 3:00 p.m. on 25th December. Example of a non-continuous interval is 9:00 a.m. to 6:00 p.m. on Mondays to Fridays in the month of March. Some researchers refer to time intervals as time expressions. We use the notation $t_i \in d$ to mean that t_i is a time instance in the time interval d.

Two time intervals can be related by any of the following relations: *disjoint*, *equality*, and *overlapping*. Two time intervals tv_i and tv_j are disjoint if the intersection of the set of time instances in tv_i with those of tv_j results in the null set. Two time intervals tv_i and tv_j are equal if the set of time instances in tv_i is equal to those of tv_j. Two time intervals tv_i and tv_j are overlapping if the intersection of the set of time instances in tv_i with those of tv_j results in a non-empty set. A special case of overlapping relation is referred to as *containment*. A time interval tv_i is contained in another interval tv_j if the set of time instances in tv_i is a subset of those in tv_j. We formally denote this as $tv_i \preceq tv_j$.

4 Relationship of Core-RBAC Entities with Time and Location

In this section, we describe how the entities in RBAC are associated with location and time. The different entities of RBAC are *Users*, *Roles*, *Sessions*, *Permissions*, *Objects* and *Operations*. We discuss how each of these components are associated with location and time.

4.1 Users

We assume that each valid user, interested in doing some location-sensitive operation, carries a locating device which is able to track his location. The location of a user changes with time. The relation $UserLocation(u,t)$ gives the location of the user at any given time instant t. Since a user can be associated with only one location at any given point of time, we have the following constraint:

$$UserLocation(u,t) = l_i \wedge UserLocation(u,t) = l_j \Leftrightarrow (l_i \subseteq l_j) \vee (l_j \subseteq l_i)$$

We define a similar function $UserLocations(u,d)$ that gives the location of the user during the time interval d. Note that, a single location can be associated with multiple users at any given point of time.

4.2 Objects

Objects can be physical or logical. Example of a physical object is a computer. Files are examples of logical objects. Physical objects have devices that transmit their location information with the timestamp. Logical objects are stored in physical objects. The location and timestamp of a logical object corresponds to the location and time of the physical object containing the logical object. We assume that each object is associated with one location at any given instant of time. Each location can be associated with many objects. The function *ObjLocation(o,t)* takes as input an object o and a time instance t and returns the location associated with the object at time t. Similarly, the function *ObjLocations(o,d)* takes as input an object o and time interval d and returns the location associated with the object.

4.3 Roles

We have three types of relations with roles. These are user-role assignment, user-role activation, and permission-role assignment.

We begin by focusing on user-role assignment. Often times, the assignment of user to roles is location and time dependent. For instance, a person can be assigned the role of U.S. citizen only in certain designated locations and at certain times only. To get the role of conference attendee, a person must register at the conference location during specific time intervals. Thus, for a user to be assigned a role, he must be in designated locations during specific time intervals. In our model, a user must satisfy spatial and temporal constraints before roles can be assigned. We capture this with the concept of *role allocation*. A role is said to be *allocated* when it satisfies the temporal and spatial constraints needed for role assignment. A role can be assigned once it has been allocated. $RoleAllocLoc(r)$ gives the set of locations where the role can be allocated. $RoleAllocDur(r)$ gives the time interval where the role can be allocated. Some role s can be allocated anywhere, in such cases $RoleAllocLoc(s) = universe$. Similarly, if role p can be assigned at any time, we specify $RoleAllocDur(p) = always$.

Some roles can be activated only if the user is in some specific locations. For instance, the role of audience of a theater can be activated only if the user is in the theater when the show is on. The role of conference attendee can be activated only if the user is in the conference site while the conference is in session. In short, the user must satisfy temporal and location constraints before a role can be activated. We borrow the concept of *role-enabling* [1,8] to describe this. A role is said to be *enabled* if it satisfies the temporal and location constraints needed to activate it. A role can be activated only if it has been enabled. $RoleEnableLoc(r)$ gives the location where role r can be activated and $RoleEnableDur(r)$ gives the time interval when the role can be activated.

The predicate $UserRoleAssign(u,r,d,l)$ states that the user u is assigned to role r during the time interval d and location l. For this predicate to hold, the location of the user when the role was assigned must be in one of the locations where the role allocation can take place. Moreover, the time of role assignment must be in the interval when role allocation can take place.

$$\begin{aligned}&UserRoleAssign(u,r,d,l) \Rightarrow (UserLocation(u,d) = l) \wedge \\ &\qquad (l \subseteq RoleAllocLoc(r)) \wedge (d \subseteq RoleAllocDur(r))\end{aligned}$$

The predicate $UserRoleActivate(u,r,d,l)$ is true if the user u activated role r for the interval d at location l. This predicate implies that the location of the user during the role activation must be a subset of the allowable locations for the activated role and all times instances when the role remains activated must belong to the duration when the role can be activated and the role can be activated only if it is assigned.

$$\begin{aligned}&UserRoleActivate(u,r,d,l) \Rightarrow \\ &\quad (l \subseteq RoleEnableLoc(r)) \wedge (d \subseteq RoleEnableDur(r)) \wedge UserRoleAssign(u,r,d,l)\end{aligned}$$

The additional constraints imposed upon the model necessitates changing the preconditions of the functions $AssignRole$ and $ActivateRole$.

The permission role assignment is discussed later.

4.4 Sessions

In mobile computing or pervasive computing environments, we have different types of sessions that can be initiated by the user. Some of these sessions can be location-dependent, others not. Thus, sessions are classified into different types. Each instance of a session is associated with some type of a session. The type of session instance s is given by the function $Type(s)$. The type of the session determines the allowable location. The allowable location for a session type st is given by the function $SessionLoc(st)$.

When a user u wants to create a session si, the location of the user for the entire duration of the session must be contained within the location associated with the session. The predicate $SessionUser(u,s,d)$ indicates that a user u has initiated a session s for duration d.

$$SessionUser(u,s,d) \Rightarrow (UserLocation(u,d) \subseteq SessionLoc(Type(s)))$$

Since sessions are associated with locations, not all roles can be activated within some session. The predicate $SessionRoles(u,r,s,d,l)$ states that user u initiates a session s and activates a role for duration d and at location l.

$$SessionRole(u,r,s,d) \Rightarrow UserRoleActivate(u,r,d,l) \wedge l \subseteq SessionLoc(Type(s)))$$

4.5 Permissions

The goal of our model is to provide more security than their traditional counterparts. This happens because the time and location of a user and an object are taken into account before making the access decisions. Our model also allows us to model real-world requirements where access decision is contingent upon the time and location associated with the user and the object. For example, a teller may access the bank confidential file if and only if he is in the bank and the file location is the bank secure room and the access is granted only during the working hours. Our model should be capable of expressing such requirements.

Permissions are associated with roles, objects, and operations. We associate three additional entities with permission to deal with spatial and temporal constraints: user location, object location, and time. We define three functions to retrieve the values of these entities. $PermRoleLoc(p,r)$ specifies the allowable locations that a user playing the role r must be in for him to get permission p. $PermObjLoc(p,o)$ specifies the allowable locations that the object o must be in so that the user has permission to operate on the object o. $PermDur(p)$ specifies the allowable time when the permission can be invoked.

We define another predicate which we term $PermRoleAcquire(p,r,d,l)$. This predicate is true if role r has permission p for duration d at location l. Note that, for this predicate to be true, the time interval d must be contained in the duration where the permission can be invoked and the role can be enabled. Similarly, the location l must be contained in the places where the permission can be invoked and role can be enabled.

$$\begin{aligned}&PermRoleAcquire(p,r,d,l) \Rightarrow (l \subseteq (PermRoleLoc(p,r) \cap RoleEnableLoc(r)))\\ &\quad \wedge (d \subseteq (PermDur(p) \cap RoleEnableDur(p)))\end{aligned}$$

The predicate $PermUserAcquire(u,o,p,d,l)$ means that user u can acquire the permission p on object o for duration d at location l. This is possible only when the permission p is assigned some role r which can be activated during d and at location l, the user location and object location match those specified in the permission, the duration d matches that specified in the permission.

$$\begin{aligned}&PermRoleAcquire(p,r,d,l) \wedge UserRoleActivate(u,r,d,l)\\ &\quad \wedge (ObjectLocation(o,d) \subseteq PermObjectLoc(p,o)) \Rightarrow PermUserAcquire(u,o,p,d,l)\end{aligned}$$

5 Impact of Time and Location on Role-Hierarchy

The structure of an organization in terms of lines of authority can be modeled as an hierarchy. This organization structure is reflected in RBAC in the form of a role hierarchy [13]. Role hierarchy is a relation among roles. This relation is transitive, and anti-symmetric. Roles higher up in the hierarchy are referred to as senior roles and those lower down are junior roles. The major motivation for adding role hierarchy to RBAC was to simplify role management. Senior roles can inherit the permissions of junior roles, or a senior role can activate a junior role, or do both depending on the nature of the hierarchy. This obviates the need for separately assigning the same permissions to all members belonging to a hierarchy.

Joshi et al. [8] identify two basic types of hierarchy. The first is the permission inheritance hierarchy where a senior role x inherits the permission of a junior role y. The second is the role activation hierarchy where a user assigned to a senior role can activate a junior role. Each of these hierarchies may be constrained by location and temporal constraints. Consequently, we have a number of different hierarchical relationships in our model.

Definition 7. [Unrestricted Permission Inheritance Hierarchy] *Let x and y be roles such that $x \geq y$, that is, senior role x has an unrestricted permission-inheritance relation over junior role y. In such a case, x inherit's y's permissions but not the locations and time associated with it. This is formalized as follows:*

$$\forall p, (x \geq y) \wedge PermRoleAcquire(p,y,d,l) \Rightarrow PermRoleAcquire(p,x,d',l')$$

In the above hierarchy, a senior role inherits the junior roles permissions. However, unlike the junior role, these permissions are not restricted to time and location. Account auditor role inherits the permissions from the accountant role. He can use the permissions at any time and at any place.

Definition 8. [Unrestricted Activation Hierarchy] *Let x and y be roles such that $x \succcurlyeq y$, that is, senior role x has a role-activation relation over junior role y. Then, a user assigned to role x can activate role y at any time and at any place. This is formalized as follows:*

$$\forall u, (x \succcurlyeq y) \wedge UserRoleActivate(u,x,d,l) \Rightarrow UserRoleActivate(u,y,d',l')$$

Here again a user who can activate a senior role can also activate a junior role. This junior role can be activated at any time and place. A project manager can activate the code developer role at any time and at any place.

Definition 9. [Time Restricted Permission Inheritance Hierarchy] *Let x and y be roles such that $x \geq_t y$, that is, senior role x has a time restricted permission-inheritance relation over junior role y. In such a case, x inherit's y's permissions together with the temporal constraints associated with the permission. This is formalized as follows:*

$$\forall p, (x \geq_{lt} y) \wedge PermRoleAcquire(p,y,d,l) \Rightarrow PermRoleAcquire(p,x,d,l')$$

In the above hierarchy, a senior role inherits the junior roles permissions. However, the duration when the permissions are valid are those that are associated with the junior roles. A contact author can inherit the permissions of the author until the paper is submitted.

Definition 10. [Time Restricted Activation Hierarchy] *Let x and y be roles such that $x \succeq_t y$, that is, senior role x has a role-activation relation over junior role y. Then, a user assigned to role x can activate role y only at the time when role y can be enabled. This is formalized as follows:*

$$\forall u,\ (x \succeq_{tl} y) \wedge UserRoleActivate(u,x,d,l) \wedge d \subseteq RoleEnableDur(y) \Rightarrow UserRoleActivate(u,y,d,l')$$

Here again a user who can activate a senior role can also activate a junior role. However, this activation is limited to the time when the junior role can be activated. A program chair can activate a reviewer role only during the review period.

Definition 11. [Location Restricted Permission Inheritance Hierarchy] *Let x and y be roles such that $x \geq_l y$, that is, senior role x has a location restricted permission-inheritance relation over junior role y. In such a case, x inherit's y's permissions together with the location constraints associated with the permission. This is formalized as follows:*

$$\forall p, (x \geq_{lt} y) \wedge PermRoleAcquire(p,y,d,l) \Rightarrow PermRoleAcquire(p,x,d',l)$$

In the above hierarchy, a senior role inherits the junior roles permissions. These permissions are restricted to the locations imposed on the junior roles. A top secret scientist inherits the permission of top secret citizen only when he is in top secret locations.

Definition 12. [Location Restricted Activation Hierarchy] *Let x and y be roles such that $x \succeq_l y$, that is, senior role x has a role-activation relation over junior role y. Then, a user assigned to role x can activate role y only at the places when role y can be enabled. This is formalized as follows:*

$$\forall u,\ (x \succeq_{tl} y) \wedge UserRoleActivate(u,x,d,l) \wedge l \subseteq RoleEnableLoc(y) \Rightarrow UserRoleActivate(u,y,d',l)$$

Here again a user who can activate a senior role can also activate a junior role. However, this activation is limited to the place where the junior role can be activated. A Department Chair can activate a Staff role only when he is in the Department.

Definition 13. [Time Location Restricted Permission Inheritance Hierarchy] *Let x and y be roles such that $x \geq y$, that is, senior role x has a time-location restricted permission-inheritance relation over junior role y. In such a case, x inherit's y's permissions together with the temporal and location constraints associated with the permission. This is formalized as follows:*

$$\forall p, (x \geq_{lt} y) \wedge PermRoleAcquire(p,y,d,l) \Rightarrow PermRoleAcquire(p,x,d,l)$$

In the above hierarchy, a senior role inherits the junior roles permissions. These permissions are restricted to time and locations imposed on the junior roles. Daytime doctor role inherits permission of daytime nurse role only when he is in the hospital during the daytime.

Definition 14. [Time Location Restricted Activation Hierarchy] *Let x and y be roles such that $x \succcurlyeq_{tl} y$, that is, senior role x has a role-activation relation over junior role y. Then, a user assigned to role x can activate role y only at the places and during the time when role y can be enabled. This is formalized as follows:*

$$\forall u,\ (x \succcurlyeq_{tl} y) \wedge UserRoleActivate(u,x,d,l) \wedge d \subseteq RoleEnableDur(y) \\ \wedge l \subseteq RoleEnableLoc(y) \Rightarrow UserRoleActivate(u,y,d,l)$$

Here again a user who can activate a senior role can also activate a junior role. However, this activation is limited to the time and place where the junior role can be activated. User who has a role of mobile user can activate the weekend mobile user role only if he/she is in the US during the weekend.

It is also possible for a senior role and a junior role to be related with both permission inheritance and activation hierarchies. In such a case, the application will choose the type of inheritance hierarchy and activation hierarchy needed.

6 Impact of Time and Location on Static Separation of Duties

Separation of duties (SoD) enables the protection of the fraud that might be caused by the user [14]. SoD can be either static or dynamic. Static Separation of Duty (SSoD) comes in two varieties. First one is with respect to user role assignment. The second one is with respect to permission role assignment. In this case, the SoD is specified as a relation between roles. The idea is that the same user cannot be assigned to the same role. Due to the presence of temporal and spatial constraints, we can have different flavors of separation of duties – some that are constrained by temporal and spatial constraints and others that are not. In the following we describe the different separation of duty constraints.

Definition 15. [Weak Form of SSoD - User Role Assignment] *Let x and y be two roles such that $x \neq y$. $x,y \in SSOD_w(ROLES)$ if the following condition holds:*

$$UserRoleAssign(u,x,d,l) \Rightarrow \neg\, UserRoleAssign(u,y,d,l)$$

The above definition says that a user u assigned to role x during time d and location l cannot be assigned to role y at the same time and location if x and y are related by $SSOD_w$. An example where this form is useful is that a user should not be assigned the audience role and mobile user role at the same time and location.

Definition 16. [Strong Temporal Form of SSoD - User Role Assignment] *Let x and y be two roles such that $x \neq y$. $(x,y) \in SSOD_t(ROLES)$ if the following condition holds:*

$$UserRoleAssign(u,x,d,l) \Rightarrow \neg\, (\exists d' \subseteq always \bullet UserRoleAssign(u,y,d',l))$$

The above definition says that a user u assigned to role x during time d and location l cannot be assigned to role y at any time in the same location if x and y are related by $SSOD_t$. The consultant for oil company A will never be assigned the role of consultant for oil company B in the same country.

Definition 17. [Strong Spatial Form of SSoD - User Role Assignment] *Let x and y be two roles such that $x \neq y$. $(x,y) \in SSOD_l(ROLES)$ if the following condition holds:*

$$UserRoleAssign(u,x,d,l) \Rightarrow \neg\ (\exists l' \subseteq universe \bullet UserRoleAssign(u,y,d,l'))$$

The above definition says that a user u assigned to role x during time d and location l, he cannot be assigned to role y at the same time at any location if x and y are related by $SSOD_l$. A person cannot be assigned the roles of realtor and instructor at the same time.

Definition 18. [Strong Form of SSoD - User Role Assignment] *Let x and y be two roles such that $x \neq y$. $(x,y) \in SSOD_s(ROLES)$ if the following condition holds:*

$$UserRoleAssign(u,x,d,l) \Rightarrow \neg\ (\exists l' \subset universe, \exists d' \subseteq always \bullet UserRoleAssign(u,y,d',l'))$$

The above definition says that a user u assigned to role x during time d and location l, he cannot be assigned to role y at any time or at any location if x and y are related by $SSOD_s$. The same employee cannot be assigned the roles of male and female employee at any given corporation.

We next consider the second form of static separation of duty that deals with permission role assignment. The idea is that the same role should not acquire conflicting permissions. The same manager should not make a request for funding as well as approve it.

Definition 19. [Weak Form of SSoD - Permission Role Assignment] *Let p and q be two permissions such that $p \neq q$. $(p,q) \in SSOD\_PRA_w$ if the following condition holds:*

$$PermRoleAcquire(p,x,d,l) \Rightarrow \neg\ PermRoleAcquire(q,x,d,l))$$

The above definition says that if permissions p and q are related through weak SSoD Permission Role Assignment and x has permission p at time d and location l, then x should not be given permission q at the same time and location.

Definition 20. [Strong Temporal Form of SSoD - Permission Role Assignment] *Let p and q be two permissions such that $p \neq q$. $(p,q) \in SSOD\_PRA_t$ if the following condition holds:*

$$PermRoleAcquire(p,x,d,l) \Rightarrow \neg\ (\exists d' \subseteq always \bullet PermRoleAcquire(q,x,d',l))$$

The above definition says that if permissions p and q are related through strong temporal SSoD Permission Role Assignment and x has permission p at time d and location l, then x should not get permission q at any time in location l.

Definition 21. [Strong Spatial Form of SSoD - Permission Role Assignment] *Let p and q be two permissions such that $p \neq q$. $(p,q) \in SSOD\_PRA_t$ if the following condition holds:*

$PermRoleAcquire(p,x,d,l) \Rightarrow \neg\,(\exists l' \subset universe \bullet PermRoleAcquire(q,x,d,l'))$

The above definition says that if permissions p and q are related through strong spatial SSoD Permission Role Assignment and x has permission p at time d and location l, then x should not be given permission q at the same time.

Definition 22. [Strong Form of SSoD - Permission Role Assignment] *Let p and q be two permissions such that $p \neq q$. $(p,q) \in SSOD\_PRA_s$ if the following condition holds:*

$PermRoleAcquire(p,x,d,l) \Rightarrow \neg\,(\exists l' \subset universe, \exists d' \subseteq always \bullet PermRoleAcquire(q,x,d',l'))$

The above definition says that if permissions p and q are related through strong SSoD Permission Role Assignment, then the same role should never be given the two conflicting permissions.

7 Impact of Time and Location on Dynamic Separation of Duties

Static separation of duty ensures that a user does not get assigned conflicting roles or a role is not assigned conflicting permissions. Dynamic separation of duty addresses the problem that a user is not able to activate conflicting roles during the same session.

Definition 23. [Weak Form of DSoD] *Let x and y be two roles such that $x \neq y$. $(x,y) \in DSOD_s$ if the following condition holds:*

$SessionRole(u,x,s,d,l) \Rightarrow \neg\, SessionRole(u,y,s,d,l))$

The above definition says that if roles x and y are related through weak DSoD and if user u has activated role x in some session s for duration d and location l, then u cannot activate role y during the same time and in the same location in session s. In the same session, a user can activate a sales assistant role and a customer role. However, both these roles should not be activated at the same time in the same location.

Definition 24. [Strong Temporal Form of DSoD] *Let x and y be two roles such that $x \neq y$. $(x,y) \in DSOD_s$ if the following condition holds:*

$SessionRole(u,x,s,d,l) \Rightarrow \neg\,(\exists d' \subset always, \bullet SessionRole(u,y,s,d',l))$

The above definition says that if roles x and y are related through strong temporal DSoD and if user u has activated role x in some session s, then u can never activate role y any time at the same location in the same session. In a teaching session in a classroom, a user cannot activate the the grader role once he has activated the student role.

Definition 25. [Strong Spatial Form of DSoD] *Let x and y be two roles such that $x \neq y$. $(x,y) \in DSOD_l$ if the following condition holds:*

$SessionRole(u,x,s,d,l) \Rightarrow \neg\,(\exists l' \subseteq universe \bullet SessionRole(u,y,s,d,l'))$

The above definition says that if roles x and y are related through strong DSoD and if user u has activated role x in some session s, then u can never activate role y in session s during the same time in any location. If a user has activated the Graduate Teaching Assistant role in his office, he cannot activate the Lab Operator role at the same time.

Definition 26. [Strong Form of DSoD] *Let x and y be two roles such that $x \neq y$. $(x,y) \in DSOD_s$ if the following condition holds:*

$$SessionRole(u,x,s,d,l) \Rightarrow \neg\ (\exists l' \subset universe, \exists d' \subseteq always \bullet SessionRole(u,y,s,d',l'))$$

The above definition says that if roles x and y are related through strong DSoD and if user u has activated role x in some session s, then u can never activate role y in the same session. A user cannot be both an code developer and a code tester in the same session.

8 Example Scenario

Example 1. Consider the following access control policy of SECURE bank

1. The organization has five users: Tom, Leena, Diana, Nina and Sam.
2. The organization consists of six roles: teller, loan officer, daytime system operator, nighttime system operator, system operator manager, auditor
3. The teller can read and write teller files only from the teller booth during working hours, that is, 9:00AM - 6:00PM, Monday to Friday.
4. The loan officer can read and write loan files only from the loan office during working hours, that is, 9:00AM - 6:00PM, Monday to Friday.
5. The daytime system operator (DTSO) can backup any file from anywhere in the SECURE bank building during working hours, that is, 9:00AM - 6:00PM, Monday to Friday.
6. The nighttime system operator (NTSO) can backup and restore any file from anywhere in the SECURE bank building during nighttime shift, that is, 6:00PM - 9:00AM, Monday to Friday.
7. The system operator manager (SOM) rights consist of all rights from daytime system operator and night time system operator.
8. The auditor can audit teller files during the working hours.
9. The same person cannot be the teller and the auditor in the same session.
10. Teller files and loan files can be written during working hours only

We can represent the above access control policy using STRBAC.

1. $WorkingHours = \{\{2,3,4,5,6\}.Days + 10.Hours \triangleright 9.Hours\}$
2. $NightTime = \{\{2,3,4,5,6\}.Days + 19.Hours \triangleright 14.Hours\}$
3. Set of Time Intervals $= \{WorkingHours, NighTime\}$
4. Set of locations $= \{TellerBooth, LoanOffice, ComputerRoom, Building\}$
5. $Users = \{Tom, Leena, Diana, Nina, Sam\}$
6. $Roles = \{Teller, LoanOfficer, Auditor, DTSO, NTSO, SOM\}$
7. *RoleEnable* = {(*Teller, WorkingHours, TellerBooth*), (*Loaner, WorkingHours, LoanerOffice*), (*DTSO, WorkingHours, Building*), (*NTSO, NightTime, Building*), (*SOM, AnyTime, Building*)}
8. *Permissions* consists of
   - *readTellerFile* = (*Read, TellerFile, AnyTime, TellerBooth, ComputerRoom*)

- $writeTellerFile$ = (*Write, TellerFile, WorkingHours, TellerBooth, ComputerRoom*)
- $readLoanerFile$ = (*Read, LoanerFile, AnyTime, LoanerOffice, ComputerRoom*)
- $writeLoanerFile$ = (*Write, LoanerFile, WorkingHours, LoanerOffice, ComputerRoom*)
- $WHBackupFile$ = (*Backup, AllFile, WorkingHours, Anywhere, ComputerRoom*)
- $NTBackupFile$ = (*Backup, AllFile, NightTime, Anywhere, ComputerRoom*)
- $NTRestoreFile$ = (*Restore, AllFile, NightTime, Anywhere, ComputerRoom*)

9. $UserAssignment$ = {(*Tom, Teller*), (*Leena, Loaner*), (*Diana, DTSO*), (*Nina, NTSO*), (*Sam, SOM*)}
10. *PermAssign* = {(*Teller, readTellerFile*), (*Teller, writeTellerFile*), (*Loaner, readLoanerFile*), (*Loaner, writeLoanerFile*), (*DTSO, WHBackupFile*), (*NTSO, NTBackFile*), (*NTSO, NTRestoreFile*)}
11. Role hierarchy $RH = (SOM \succsim_{s,tl} DTSO) \wedge (SOM \succsim_{s,tl} NTSO)$
12. $DSOD_s = (Teller, Auditor)$

This is just one possible way to model the requirements. Other models are possible as well.

9 Conclusion and Future Work

Traditional access control models do not take into account environmental factors before making access decisions. Such models may not be suitable for pervasive computing applications. Towards this end, we proposed a spatio-temporal role based access control model. We identified the entities and relations in RBAC and investigated their dependence on location and time. This dependency necessitates changes in the invariants and the operations of RBAC. The behavior of the model is formalized using constraints.

A lot of work remains to be done. One is the analysis of the model. We have proposed many different constraints. We need to understand the interaction of these constraints and the different types of relationships between them. Specifically, we are interested in finding conflicts and redundancies among the constraint specification. Such analysis is needed before our model can be used for real world applications. We plan to investigate how to automate this analysis. We also plan to implement our model. We need to investigate how to store location and temporal information and how to automatically detect role allocation and enabling using triggers. Once we have an implementation, we validate our model using some prototype application.

Acknowledgement

This work was supported in part by AFOSR under contract number FA9550-07-1-0042.

References

1. Bertino, E., Bonatti, P.A., Ferrari, E.: TRBAC: a temporal role-based access control model. In: RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control, pp. 21–30. ACM Press, New York, NY, USA (2000)
2. Bertino, E., Catania, B., Damiani, M.L., Perlasca, P.: GEO-RBAC: a spatially aware RBAC. In: SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies, pp. 29–37. ACM Press, New York, NY, USA (2005)
3. Chandran, S.M., Joshi, J.B.D.: LoT-RBAC: A Location and Time-Based RBAC Model. In: WISE, pp. 361–375 (2005)
4. Covington, M.J., Fogla, P., Zhan, Z., Ahamad, M.: A Context-Aware Security Architecture for Emerging Applications. In: Proceedings of the Annual Computer Security Applications Conference, Las Vegas, NV, USA, pp. 249–260 (December 2002)
5. Covington, M.J., Long, W., Srinivasan, S., Dey, A., Ahamad, M., Abowd, G.: Securing Context-Aware Applications Using Environment Roles. In: Proceedings of the 6th ACM Symposium on Access Control Models and Technologies, pp. 10–20. Chantilly, VA, USA (May 2001)
6. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and Systems Security 4(3) (August 2001)
7. Hengartner, U., Steenkiste, P.: Implementing Access Control to People Location Information. In: Proceeding of the SACMAT'04 Yorktown Heights, California, USA (June 2004)
8. 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)
9. Leonhardt, U., Magee, J.: Security Consideration for a Distributed Location Service. Imperial College of Science, Technology and Medicine, London, UK (1997)
10. Ray, I., Kumar, M.: Towards a Location-Based Mandatory Access Control Model. Computers & Security 25(1) (February 2006)
11. Ray, I., Kumar, M., Yu, L.: LRBAC: A Location-Aware Role-Based Access Control Model. In: Proceedings of the 2nd International Conference on Information Systems Security, Kolkata, India, pp. 147–161 (December 2006)
12. 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, pp. 343–352 (December 2002)
13. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Computer 29(2), 38–47 (1996)
14. Simon, R., Zurko, M.E.: Separation of duty in role-based environments. In: CSFW '97: Proceedings of the 10th Computer Security Foundations Workshop (CSFW '97), Washington, DC, USA, pp. 183–194. IEEE Computer Society Press, Los Alamitos (1997)
15. Yu, H., Lim, E.-P.: LTAM: A Location-Temporal Authorization Model. In: Jonker, W., Petković, M. (eds.) SDM 2004. LNCS, vol. 3178, pp. 172–186. Springer, Heidelberg (2004)

Towards a Times-Based Usage Control Model

Baoxian Zhao[1], Ravi Sandhu[2], Xinwen Zhang[3], and Xiaolin Qin[4]

[1] George Mason University, Fairfax VA, USA
bzhao@gmu.edu
[2] Institute for Cyber-Security Research, Univ. of Texas at San Antonio, USA
ravi.sandhu@utsa.edu
[3] Samsung Information Systems America, San Jose, CA, USA
xinwen.z@samsung.com
[4] Nanjing University of Aeronautics and Astronautics, Nanjing, China
qinxcs@nuaa.edu.cn

Abstract. Modern information systems require temporal and privilege-consuming usage of digital objects. To meet these requirements, we present a new access control model–Times-based Usage Control (TUCON). TUCON extends traditional and temporal access control models with times-based usage control by defining the maximum times that a privilege can be exercised. When the usage times of a privilege is consumed to zero or the time interval of the usage is expired, the privilege exercised on the object is automatically revoked by the system. Formal definitions of TUCON actions and rules are presented in this paper, and the implementation of TUCON is discussed.

Keywords: Access Control, Usage Control, Times-based Usage Control, TUCON, Authorization.

1 Introduction

The rapid development in information technology, especially in electronic commerce applications, requires additional features for access control. In recent information systems, usage of a digital object can be not only time-independent like read and write, but also temporal and times-consuming, such as payment-based online reading metered by reading times or chapters, or a downloadable music file that can only be played 10 times. In these applications, the access to an object may decrease, expire, or be revoked along with the usage times of the object.

Traditional and temporal access control models are not suitable for the above requirements since the authorization decisions in these models are generally made at the requested time but hardly recognize ongoing controls for times constrained access or for immediate revocation. In order to meet these requirements in modern access control, this paper presents a new access control model, called Times-based Usage Control (TUCON).

Compared to traditional models, TUCON features with the usage times of privileges and valid periods for usage of digital objects, which enable the ability to express consumed privileges and define their period constraints. The usage times can be triggered by active tasks in TUCON model. For example, once an Internet user pays $10 for an on-line music system, he can enjoy 10 times on-line listening privilege. When a subject is accessing the object in TUCON, the usage times of this subject is decreased by 1. If the times is consumed to zero or the time interval of usage is expired, authorization for the subject is revoked. With the decreasing of usage times, authorizations can be updated during the whole access process, and transferred among users without the problem of privilege chains faced by current and traditional access control. Compared to authorizations in traditional and temporal access control models, authorizations in TUCON are mutable and flexible.

With usage times constraints in TUCON, system resources and privileges can be prevented from being abused. It's known that, through occupying too many resources, some worms and viruses can attack computer systems, such as Code Red [20] and Dukes [17]. If reasonable usage times are given for system resources, such kinds of attacks can be avoided to a great extent [11].

The paper is organized as follows: In Section 2, some related work are discussed. Section 3 shows a simple motivating example, which traditional and temporal access control models cannot support well. Section 4 presents the temporal and times-based constraints in the proposed TUCON model. Authorizations and authorization rules in TUCON are also discussed in this section. In Section 5, we give the implementation of TUCON in practice. Section 6 concludes this paper and presents our future work.

2 Related Work

The development of access control models has experienced a long history. There are two main approaches in this field. One is about traditional access control models. This approach is the earliest research work for access control and originates from the research of discretionary access control (DAC) [2,6,12,21]. A classic paper by Lampson [2] introduced this basic ideas. Because DAC has an inherent weakness that information can be copied from one object to another, it is difficult for DAC to enforce a safety policy and protect against some security attacks. In order to overcome the shortcoming of DAC, mandatory access control (MAC) was invented to enforce lattice-based confidentiality policies [4,5] in the face of Trojan Horse attacks. MAC does not consider covert channels, but covert channels are expensive to eliminate [22]. Sandhu et al presented Role-based access control (RBAC) [23], which has been considered as a promising alternative to DAC and MAC. There have been much progress in traditional access control, but its core has largely remained unchanged and centered around the access matrix model [2,3].

The other approach is about the research of temporal access control models, which introduce the temporal attributes into traditional access control with temporal logic. The approach is based on traditional access control. A temporal authorization model was first proposed by Bertino et al in [7], which is based on temporal intervals of validity for authorization and temporal dependencies among authorizations. In [8,9], Bertino et al extended the range of temporal intervals to temporal periods and suggested an access control model supporting periodicity constrains and temporal reasoning. Following the RBAC model, the Temporal-RBAC (TRBAC) model was presented in [10], which supports temporal dependencies among roles. At the same time, Avigdor et al suggested another authorization model for temporal data called Temporal Data Authorization Model (TDAM) [1]. TDAM extended the basic authorization model by facilitating it with the capability to express authorizations based on the temporal attributes associated with data, such as transaction time and valid time. Recently, TRBAC is extended to the Generalized Temporal RBAC (GTRBAC) in [13], which enables RBAC to express a wider range of temporal constraints. All these temporal access control models primary consider authorization decisions constrained by certain time periods.

Although many researchers in the field of access control have made great contributions to the progress of access control, authorizations in these models are still static authorization decisions based on subjects' permissions on target objects. Once an access to the object is permitted, the subject can access it repeatedly at the valid time intervals.

Sandhu , Park et al have proposed the Usage Control (UCON) [14,15,24,27,28] model to solve these problems. The UCON model considers this temporal and consumed attributes as the mutable attributes of subjects or objects [15]. The UCON model has unified traditional access control models and temporal access models with its ABC (Authorizations, oBligations and Conditions) [14] core models.

Our approach to solve these problems is different from the work of the UCON model. In TUCON, we focus on periods and usage times of accessing rather than a single access in UCON. In other words, usage times in TUCON is a sequence of accesses. Temporal and consumed authorizations are enforced in TUCON, which make access control simple and easy to be implemented. However, until now, UCON hasn't been put into the practice because the administration of authorizations in UCON is potentially complex and difficult to implement. Therefore, to meet new requirements in the modern access control, TUCON is suitable for applications in the real world.

3 Motivating Example

In this section, a simple example is given to motivate the new features of TUCON. Current and traditional access models have difficulties, or lack flexibility to specify policies in this application.

Consider a simple application with times constrained usage of digital objects, where a registered user can enjoy on-line music for 10 times if only he/she has prepaid $10. Privileges for using objects can be transferred between two registered users and their privileges will be revoked in the following two situations:

1. revocation by usage times: the usage times is zero during ongoing usage of digital objects.
2. revocation by time interval: the time interval of authorizations is expired.

Based on this policy, three different attributes are required to meet these authorizations:

1. The time interval. It includes the starting time and the end time. An access is permitted when the usage times for digital objects is more than zero at the starting time. Otherwise, at the end time, the usage privilege for using objects is revoked.
2. The valid period. Only during the valid period of usage, an access to an object can be permitted, otherwise, denied.
3. Usage times. It is the maximum value which restricts a subject accessing the object. When the usage times of an object is zero, the subject is prohibited to access this object and the privilege is automatically revoked by the system.

Through the above analysis, we give the state transition of times-based usage control actions in Figure 1.

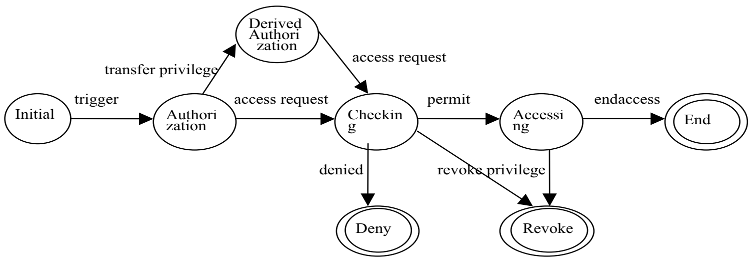


Fig. 1. The state transition of times-based usage control actions

The states and actions in Figure 1 are explained below.

1. Initial: the initial state of the system.
2. Triggers: triggering authorization of systems, which are requirements that must be satisfied for granting access. In this example, triggers are the actions that a user must register himself in our system and pre-pay $10 before enjoying this on-line service. Triggers are abstracted into logic expressions in TUCON.
3. Authorization: granting privileges of service to users if users meet authorization requirements of the system.
4. Transfer privilege: one can transfer his privilege of an object to another by decreasing his usage times.
5. Derived authorization: authorizations are derived from transferring privileges or exercising privileges.
6. Access request: the user requests to access digital objects. In section 5, the formalization of an access request is given.

7. Checking: checking the usage times and the period of the authorization .
8. Permitted and denied: If the usage times is more than zero and the time interval is not expired during the valid period , an access to digital objects is permitted, otherwise, denied.
9. Accessing: During this state, subjects are accessing digital objects. During the state of accessing, the usage times of subjects need to be updated by decreasing 1.
10. Revoke privilege and endaccess: If the usage times is not zero and the usage period is till valid after accessing, no updating is done by the system. This case is the action of endaccess. Otherwise, the system will revoke some subjects' privileges and update some authorizations. This process is the action of revoke privilege. The details of revoke privilege are introduced in Section 5.
11. Deny, Revoke and End: three final states. Deny is the sate of refusing to access without revoking privileges. Revoke is the finial state after the action of revoke privileges, while End is the one after the action of endaccess.

From the analysis of states and actions in TUCON, it is obvious that an access is not a simple action, which consists of a sequence of actions and active tasks from a subject and the system. During the whole access process, authorizations need to be updated. These requirements are far out of the scope that traditional and temporal access models can deal with. In the following, TUCON is introduced to solve these problems.

4 TUCON Model

TUCON consists of two aspects: times-based authorizations and authorization rules. Before these aspects are discussed, some preliminaries are given.

4.1 Preliminaries

In order to keep the generality of TUCON and protect information of different data models, no basic data assumption for TUCON is made here. Therefore it is easy to apply TUCON to other data models.

Assume that U denotes the set of subjects (users), O set of objects, P set of privileges for objects, $\mathbb{N}$ set of natural numbers, and T set of time intervals with a total order relation $\leq$.

Definition 1 (Periodic Expression [9]). *A periodic expression is defined as* $Q = \sum_{i=1}^{n} O_i \ .C_i \triangleleft .C_d$*, where* $O_1 \in 2^{\mathbb{N}} \cup \{all\}$*,* $O_i \in 2^{\mathbb{N}}, i = 2, \ldots, n, C_i$ *and* C_d *are calendars for* $i = 2, \ldots, n, C_d \subseteq C_n$*, and* $d \in \mathbb{N}$*.*

Let D present the set of all valid periods, then $Q \in D$. Table 1 illustrates a set of periodic expressions and their meanings. In order to simplify the following discussion, all authorization tuples are given with the same privilege on the same object, having the same time interval, and the same period. Since the implementation of operations for periodical data [9,16,18] is not the focus of this paper, any further discussion is not given in this paper.

Definition 2 (Times). *Times is a set of natural numbers, formally defined as* $\{pt \in \mathbb{N}\}$.

4.2 Authorizations

TUCON allows us to express times-based authorizations. That is, authorizations for a user to access an object in specific time intervals are specified by a periodic expression, as well as determined by times of privilege usage. Moreover, the usage times of a privilege is a natural number associated with each authorization, and a time interval is also associated with each authorization, imposing lower and upper bounds to the potentially infinite set of instants denoted by the periodic expression. We refer to an authorization together with its usage times as a *times authorization.*

Definition 3 (Times Authorization). *A times authorization is a 6-tuple (pt, s, o, priv, pn, g), where* $pt \in \mathbb{N}, s, g \in S, o \in O, priv \in P, pn \in \{+, -\}$.

Tuple (*pt, s, o, priv, pn, g*) states that user s has been authorized (if pn = '+') or denied (if pn = '-') for pt times privilege $priv$ on object o by user g. For example, the tuple *(6, Tom, Sun, read, +, Sam)* denotes that *Sam* authorizes 6 times privilege *read* on the book *Sun* to *Tom.*

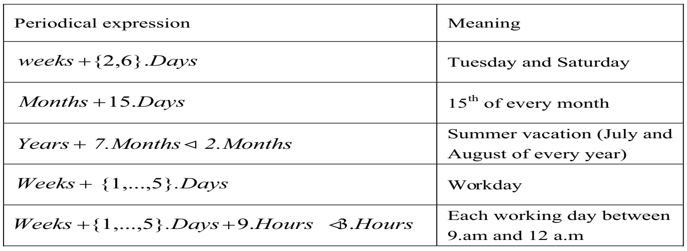
For convenience, the symbol σ is used to project some appointed area of a tuple. For example, with the tuple *A=(pt1, s1, o1, priv1, pn1, g1)*, $\sigma_A(s) = s1$ denotes the s area of the times authorization A, and $\sigma_A(s, g) = (s1, g1)$ denotes a 2-tuple consisting of s and g areas.

In TUCON, all authorizations are uniformly authorized by the system. When transferring privileges, the system can still be regarded as user g, who transfers privileges to other users, since usage times of this user g are correspondingly decreased. Consumed times reduces the transferring capability during transferring privileges. So revoking privileges, we only need to delete the privileges in our system, which doesn't have problems caused by transferring privileges in the current and traditional access control models such as cascading.

Under some conditions, privileges on objects without times constrains are needed. This kind of authorizations is referred as *non-times authorizations.*

Definition 4 (Non-Times Authorization). *When pt = -1 in a times authorization tuple, we call this times authorization as non-times authorization.*

Table 1. Example of periodic expressions



Periodical expression	Meaning
$weeks + \{2,6\}.Days$	Tuesday and Saturday
$Months + 15.Days$	15th of every month
$Years +\ 7.Months \triangleleft\ 2.Months$	Summer vacation (July and August of every year)
$Weeks +\ \{1,\ldots,5\}.Days$	Workday
$Weeks + \{1,\ldots,5\}.Days + 9.Hours\ \ \triangleleft 3.Hours$	Each working day between 9.am and 12 a.m

Notice that in TUCON, when *pn* = *'-'* in an authorization tuple, it states that this authorization is revoked, even though the usage times may not be zero.

Definition 5 (Times-Based Authorization). *A times-based authorization is a 3-tuple* (time, period, auth)*, where time represents a time interval* $[t_a, t_b]$*,* $0 \leq t_a \leq t_b \in T$*, period is a periodical expression, and auth is a 6-tuple authorization.*

A 3-tuple ($[t_a, t_b]$,*d* ,(*pt*, *s*, *o*, *priv*, *pn*, *g*)) states that user *s* has been authorized (if *pn* = '+') or denied (if *pn* = '-') for pt times privilege *priv* on object *o* by user *g* in the time interval $[t_a, t_b]$ of the period *d*. When *pt*= '-1', TUCON can be reduced to the models discussed in [7,8,9].

For a times-based authorization ([1/12/2001 ,12/24/2005], *Weeks+2.days*, *(6, Tom, file, read, +, Sam)*), it means that, between Jan. 12 , 2001 and Dec. 24, 2005, *Tom* has 6 times privilege *read* on object *file*, but he can operate this privilege only on Tuesday each week.

4.3 Authorization Rules

In this section, authorization rules, with similar semantic as [27], are introduced to organize authorizations. We start with the following predicate symbols.

1. A ternary predicate symbol, *access*, an authorization token. The first area of *access* is a time interval *time*, the second is a periodical expression *period*, and the third is a 6-tuple authorization *auth*. The predicate *access* represents authorizations explicitly inserted by the administrator.
2. A ternary predicate symbol, *deraccess*, with the same semantic meaning as *access*. The predicate *deraccess* represents authorizations derived by the system using logical rules of inference.
3. A ternary predicate symbol, *force_ access*, with the same semantic meaning as *access*. The predicate *force_ access* represents authorizations that hold for each subject on each object. It enforces the conflict resolution policy.
4. A symbol $L_i (0 \leq i \leq n)$. It can represent all the above predicate symbols and also trigger expressions required by the system.

First, a grant rule is given to express how to grant subjects authorizations.

Definition 6 (Grant Rule). *A grant rule is defined as the form of:*
access (time, period, auth) $\leftarrow L_1 \& \ldots \& L_n$

where L_i is a trigger condition expression. This expression can be developed from specific requirements for the usage of digital objects. These conditions may be triggers of some active tasks. All these depend on the requirements of the system.

Grant rules are specified to permit accesses to subjects. Whether this rule is true or not is decided by whether or not the condition expressions are satisfied. Note that in TUCON, an authorization is only permitted to grant a positive one (*pn* = '+'), not a negative one (*pn* = '-').

Example 1. In an application system *Business_system*, if a registered user *Bob* pre-pays \$1000, he can enjoy a certain super-value service *m* for 6 times during every Friday since the time 09/12/2006. Let this privilege be *super*. This authorization can be expressed by a grant rule as the following:
access([09/12/2006,+ ∞] *, Weeks+5.days, (6, Bob , m, super, +, Business_system))* $\leftarrow$ *prepay(Bob,1000)* & *register(Bob)*

Here *prepay(Bob, 1000)* means Bob pre-pays \$1000 and *register(Bob)* Bob is a registered user. Both of them are trigger condition expressions.

Definition 7 (Derived Rule). *A derived rule is defined as the form of: deraccess (time, period, auth)* $\leftarrow L_1\& \ldots \& L_n$
where L_i can be *access with conditional expressions.*

Derived rules can be used to update usage times during ongoing usage control. When an access is performed, the usage times is decreased by 1. However, derived rules only update times authorizations rather than non-times authorizations after accessing, which is discussed in resolution rules,

Derived rules also support transferring times-based authorizations. In TUCON, transferring authorizations means consuming the usage times of privileges on digital objects. When the usage times of a privilege decreases to zero, the privilege is automatically revoked by the system.

Example 2. Now Bob wants to transfer 3 times for enjoying the service m to another user Alice. These can be defined with derived rules as the following:
deraccess([09/12/2006,+ ∞] *, Weeks+5.days, (3, Alice , m, super, +, Business_system))* $\leftarrow$*access (* [09/12/2006,+ ∞] *, Weeks+5.days, (6, Bob , m, super, +, Business_system))* & *give(3, Alice, m, super, Bob)* & *less(3,6)*
deraccess([09/12/2006,+ ∞] *, Weaks+5.days, (3, Bob , m, super, +, Business_system))* $\leftarrow$*access (* [09/12/2006,+ ∞] *, Weeks+5.days, (6, Bob , m, super, +, Business_system))* & *give(3, Alice, m, super, Bob)* & *less(3,6)*

where *give(3, Alice, m, super, Bob)* states that Bob transfers 3 times of privilege *super* to Alice. *less(m,n)* states true if *m* is less than *n*.

Through above discussion, we can notice that multiple times authorizations can be given for a subject to access the same object through applying grant and derived rules. After transferring authorizations or accessing, the usage times of this authorization can be decreased to zero. So we need a rule to resolve these conflicts. A resolution rule, given below, forces a final unambiguous decision to be made.

Definition 8 (Resolution Rule). *A resolution rule is defined as the form of: force_ access (time, period, auth)* $\leftarrow L_1\& \ldots \& L_n$

where L_i can be *access or deraccess or condition expressions.* A resolution rule states that a given subject must be allowed/forbidden to perform a privilege on an object. Compared to grant and derived rules, which may cause authorizations conflicts, resolution rules state which authorizations the system must consider valid for each subject, on the basis of the existing granted or derived authorizations.

Resolution rules can be used to revoke authorizations, combine multiple times authorizations, update non-times authorizations after accessing and solve conflicts caused by times and non-times authorizations coexisting for a subject.

Example 3. In example 2, if Alice has 4 times *super* right on service *m*, a resolution rule should be used to solve conflicts after Bob transfers rights to Alice:
force_ access([09/12/2006,+ ∞] *, Weaks+5.days, (7, Alice , m, super, +, Business_system))* $\leftarrow$*access (* [09/12/2006,+ ∞] *, Weeks+5.days, (4, Alice , m, super, +, Business_system))* & *deraccess (* [09/12/2006,+ ∞] *, Weeks+5.days, (3, Alice , m, super, +, Business_system))*

If Alice has non-times right on service m, this resolution rule should be used:
force_ access([09/12/2006,+ ∞] *, Weeks+5.days, (-1, Alice , m, super, +, Business_system))* $\leftarrow$*access (* [09/12/2006,+ ∞] *, Weeks+5.days, (-1, Alice , m, super, +, Business_system))* & *deraccess (* [09/12/2006,+ ∞] *, Weeks+5.days, (3, Alice , m, super, +, Business_system))*

If Bob's has 0 times right after transferring rights, another resolution rule is added to revoke Bob's authorization:
force_ access([09/12/2006,+ ∞] *, Weeks+5.days, (0, Bob , m, super, -, Business_system))* $\leftarrow$*deraccess (* [09/12/2006,+ ∞] *, Weeks+5.days, (0, Bob , m, super, +, Business_system))* .

Depending on different situations, an administrator of the system can make a forced authorization decision by this rule. For example, when a security administrator notices that a user often sends many access requests without using services, this administrator may take actions on this user to prevent denial of service (DoS), such as revoking his authorization. Different applications have different considerations for administrators. However, as a general model, TUCON does not take any specific application into consideration. All these can be abstracted into condition expressions. Note that the conditions in resolution rules are factors which violate the security policy of systems, while those in grant rules are requirements, which must be satisfied for granting privileges.

Based on above given authorization rules, a set of rules are introduced to enforce the policy in TUCON. In the following, we just write the tuple of *auth* instead of the tuple of *(time, period , auth)*, based on the assumption stated in Section 4.1.

First, some symbols are explained to express rules as follows.

- *s, s1, system* $\in S$, where *system* is considered as the administrator and every user is a legal (registered) one in the system;
- *priv* $\in P$;
- $o \in O$;
- $Tr_i(i \in \mathbb{N})$ is the logic expression of a trigger;
- *give(s, s1, o, priv, pt)*: indicating that user s gives his *pt* usage times of privilege *priv* on object o to user *s1*;
- *accessing(s, o, priv)*: indicating that user s is performing operation privilege *priv* on object o;

– *expired(current_t)*: indicating that the current time is expired for the valid time interval, $current\_ t \in T$.

R1 *auth(pt, s, o, priv, +, system)* $\leftarrow Tr_1 \& \ldots \& Tr_n$ $(0 \leq i \leq n)$
R2a *auth(pt1, s1, o, priv, +, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *give(s, s1, o, priv)* & *(pt $\geq$ pt1)*
R2b *auth(pt-pt1, s, o, priv, +, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *give(s, s1, o, priv)* & *(pt $\geq$ pt1)*
R3a *auth(pt-1, s, o, priv, +, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *accessing(s, o, prv)*& *pt > 0*
R3b *auth(-1, s, o, priv, +, system)* $\leftarrow$ *auth(-1, s, o, priv, +, system)* & *accessing (s, o, prv)*
R4 *auth(pt+pt1, s, o, priv, +, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *auth(pt1, s, o, priv, +, system)* & *(pt $\geq$ 0)* & *(pt1 $\geq$ 0)*
R5 *auth(-1, s, o, priv, +, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *auth(-1, s, o, priv, +, system)*
R6 *auth(0, s, o, priv, -, system)* $\leftarrow$ *auth(0, s, o, priv, +, system)*
R7 *auth(pt, s, o, priv, -, system)* $\leftarrow$ *auth(pt, s, o, priv, +, system)* & *expired(current_ t)*

– Rule R1, a grant rule, says that if user satisfies requirements Tr_i before allowing access, he/she can get *pt* times for operating privilege *priv* on object o from system.
– Rule R2a, R2b, two derived rules, implement that user can give his *pt1* usage times of privilege to user *s1*.
– Rule R3a, a derived rule, says that when user *s* with a times authorization is accessing object *o*, his usage times should be decreased by 1. Rule R3b, also a resolution rule, says that when user *s* with a non-times authorization is accessing object *o*, there is no need to update his authorization.
– Rule R4, a resolution rule, says that when there exist two times-based authorizations with the same user *s* for the same privilege *priv* on the same object, usage times should be added between these authorizations to make the final authorization decision. This rule is to solve authorizations from grant rules and derived authorization rules.
– Rule R5, a resolution rule, solves the conflicts between a *non-times authorization* and a *times authorization* for the same subject, object and privilege. When there exists the above case, a *non-times authorization* to the subject will be granted.
– Rule R6, a resolution rule, says that if usage times is zero, this authorization will be revoked.
– Rule R7, a resolution rule, says that if the valid time is expired, this authorization will be revoked.

4.4 Completeness

A set of the above rules can preserve the completeness property of the policy in TUCON.

Theorem 1 (Completeness). *The policy in TUCON can be specified by a non-empty set of TUCON rules.*

Proof. If we can prove that there is no conflict decision by using these rules and these rules specify all possible decisions during the usage process in TUCON, the completeness of the policy in TUCON is guaranteed.

(1) no conflict decisions

By construction of these rules, resolution rules can be used to resolve the conflictions caused by grant rules and derived rules. After using resolution rules, there are no conflict access decisions since the resolution rule states that a subject must be allowed/forbidden to performance a privilege on an object. So we can safely conclude that there is no conflict decision made by using these rules.

(2) specifying all possible decisions

If the system state transitions in TUCON satisfy these rules, it can conclude that these rules specifies all possible decisions during the usage process.

Based on Figure 1 in Section 2, the state transitions are constructed with the following steps and illustrated in Figure 2.

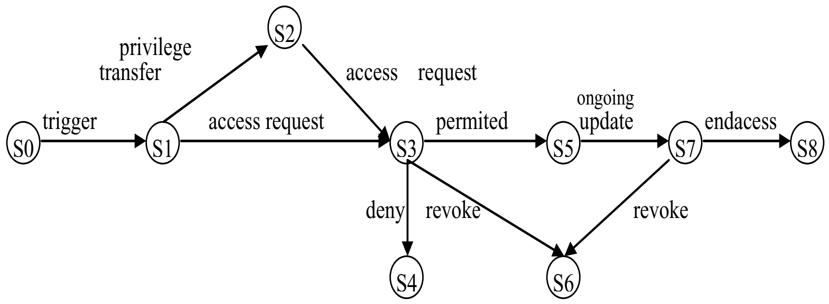


Fig. 2. State transitions

1. Initially the system state is *S0*. In the state *S0*, the subject *s* performs some triggers satisfying with *R1*. Then *s* gets a new authorization and the system state changes to *S1*.
2. In *S1*, if the subject *s* transfers some privilege to another subject, the system state arrives *S2*. During changing of states, we can use *R2a and R2b* to transfer privileges. When causing some conflict authorizations, *R4* or *R5* can be used to resolve them.
3. In *S2* or *S1*, when receiving an access request, the system state changes to *S3*.
4. In *S3*, access requirements are satisfied, the access is permitted and the new system state is *S5*. If usage times is zero, *R6* is used to revoke this privilege and new system state is *S6*. If the valid time is expired, *R7* is used to revoke this privilege and new system state is *S6*; Otherwise, the access is denied and the system state changes to *S4*. Notice that in *S4*, the privilege is not revoked by system.

5. In *S5*, after using *R3a and R3b* to update authorizations, the system new state is *S7*.
6. In *S7*, if usage times is zero, we revoke the privilege with *R6* and arrive at the state *S6*. Otherwise, the system state is *S8* after ending an access.

With simple model checking, we can verify that all the rules are satisfied in these state transitions. That is, these rules specify all possible decisions in TUCON.

Considering the two above factors, the policy in TUCON can be specified by a non-empty set of the TUCON rules. □

5 Implementation of TUCON

In the above section, TUCON has been discussed in detail. Now, the implementation of TUCON in practice is given, which includes administration of authorizations and implementation of access control.

5.1 Administration of Authorizations

In the implementation of access control models, the most important thing is administration of authorizations. All authorizations in TUCON are derived from grant, derived, and resolution rules. A set of authorizations is called a Times-based Authorization Base (TAB). A TAB includes authorizations from access, deraccess, and force_ access which are not conflict with each other.

In a TAB, operations of authorizations are to grant/revoke times and non-times authorizations to/from users. In order to support these operations, the following problems must be solved:

(a) How to deal with a situation when times and non-times authorizations co-exist for a given object, subject, and privilege?

(b) How to deal with multiple times authorizations for a given object, subject, and privilege?

We can deal with the above problems easily with the above Rule *R4* and Rule *R5*.

Next we discuss operations for revoking privileges. It includes two kinds of manipulations: repeal non-times authorizations and automatically revoke times-based authorization. First, we check an authorization tuple $au = ([t_a, t_b],\ d,$ *(pt, s, o, priv, pn, g)*) with respect to the current time. If the current time $current\_time > t_b (current\_time \in T)$ then set pn = '-' in the 6-tuple of authorization; If $t_a \leq current\_time \leq t_b$, tuple au should be further checked to make sure whether it is a times-based authorization or not, and then if $\sigma_{au}(pt) = 0$ set pn = '-' in the 6-tuple authorization au. Finally, we delete the tuples with pn = '-' in TAB to form the new TAB. Otherwise, TAB will not be changed.

5.2 Access Control

After a subject is granted with an authorization, an access request is needed to access the object.

Definition 9 (Access Request). *An access request is a 5-tuple* (t, p, s, o, priv) *where $t \in T$ is the time when the access is requested, $p \in Q$ the current period point, $s \in S$ the user who requires the access, $o \in O$ the object to be access, and $priv \in P$ a privilege exercised on object* o*.*

As far as an authorization is concerned, the first step is to judge whether this authorization is valid or not. So every access request is checked against the current TAB to determine whether the access is authorized, which is checked by the valid authorization function.

Definition 10 (Valid Authorization Function). *The valid authorization function is used to judge whether the current authorization is valid. It can be expressed as the following:*

$$G(r) = \begin{cases} au & au \in TAB \wedge \sigma_r(t) \in \sigma_{au}(time) \\ & \wedge \sigma_{\sigma_{au}(auth)}(s, o, piv) = \sigma_r(s, o, priv) \\ \emptyset & others \end{cases}$$

where r is an access request. *G(r)* returns an authorization tuple. When it is $\emptyset$, the authorization is illegal, otherwise is a legal authorization.

However, a valid authorization is not enough for an access request. The specific period of the current request access should also be checked, which is valid or not according to the period constraint of an authorization. A valid authorized access request is a request for which an authorization exists in the current TAB, which is checked by the following valid access function.

In order to express the definition of a valid access function conveniently, a useful expression is given for the relationship between a period point and the period. If a specific period point $p \in Q$ belongs to the period $per \in Q$, it is denoted as $p = \prod(per)$.

Definition 11 (Valid Access Function). *The valid access function is used to judge whether the access request is valid according to the current TAB. It can be expressed as follows:*

$$F(r) = \begin{cases} true & \exists G(r)(\sigma_r(p) = \prod(\sigma_{G(r)}(period))) \\ false & others \end{cases}$$

where r is an access request. If *F(r)* is *true*, the access is valid.

After a subject submits an access request r and *F(r)* is *true*, this subject is permitted to access the object. During the following process of accessing, there are three kinds of situations that should be considered. If a requested authorization tuple is a non-times authorization, the TAB remains unchanged. If it is a times authorization, when the times is more than zero, the number of the privilege times is subtracted by 1, and then the TAB by resolution rules is modified. If the number of times is zero, we delete this tuple from the TAB. The concrete algorism is briefly described in the following, where the TAB is current times-based authorization base, and r is an access request.

```
Access_ control(TAB, r)
{......
  $au = G(r)$;        // use the valid authorization to return a set function of
                      // authorization tuple, and then judge whether the
                      //authorization is valid
  if ( $au = \emptyset$ )
            error("Illegal Authorization ");
  if ($\sigma_{au}(pt) = 0$) // judge whether the privilege times is 0, in order
                      // to decide whether the authorization should be revoked or not.
      { revoke the authorization set pn ='-' in the 6-tuple
        of authorization, and delete those derived authoriza-
        tion tuples by resolution rules with pn ='-1', and
        form the new TAB;
        error ("This authorization does not exist ");
      }
  $k = F( r )$;       // use the valid access function to return a
                      // boolean value, with which to judge
                      // whether the access is valid.
  if   ( $k$ = false )
            error ("Illegal Access ! ");
  $i = \sigma_{\sigma_{au}}(pt)$;       // the times of privilege.
   if ( $i > 0$)
     { get by subtracting 1 from au's privilege times,
       do some modifications in the TAB and then get
       the new TAB
     }
     ......
  }
```

6 Conclusion and Future Work

To meet new requirements in recent information systems, this paper presents a new access control model— the Times-based Usage Control (TUCON) Model, TUCON supports both consumable and temporal authorizations, and persistent authorizations with transferring authorizations. The key concept of TUCON is the usage times of privileges, which makes the implementation of access control more active and mutable, and protects resources from being abused.

TUCON has a vast range of applications in modern information society and can effectively solve the problems of consumed privileges in the validity of the period, especially in the times-metered systems and electronic commerce systems.

TUCON can be viewed as one of the specific research problems for mutable attributes [15] in modern access control. In order to clearly express our key point of this model, we analyze TUCON with different usage times, just assuming that the same privileges on the same digital objects have the same time interval and the same period. It is an interesting research topic to consider the different time intervals and different periods in TUCON. This research work is currently under our way. Based on the current progress of TUCON, development of the administration of authorizations in UCON is also future research work.

References

1. Gal, A., Atluri, V.: An Authorization Model for temporal Data. ACM Transactions on Information and System Security 5(1) (Feburary 2002)
2. Lampson, B.W.: Protection. 5th Princeton Symposium on Information Science and Systems, 1971. Reprinted in ACM Operating Systems Review, 8(1), 18-24 (1974)
3. Landwehr, C.: Protection (Security) Models and Policy. In: The Computer Science and Engineering Handbook, pp. 1914–1928. CRC Press, USA (1997)
4. Bell, D.E., Lapadula, L.J.: Secure computer systems: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306,The Mitre Corporation, Bedford, MA (March 1975)
5. Denning, D.E.: A lattice Model of secure information flow. Communications of ACM. 19(5), 236–243 (1976)
6. Downs, D.D., Rub, J.R., Kung, K.C, Jordan, C.S.: Issues in discretionary access control. In: In the procceding of IEEE Symposium on Research in Security and Privacy, pp. 208–218. IEEE Press, NJ, New York (April 1985)
7. Bertino, E., Bettini, C., Samarati, P.: A Temporal Authorization Model. CCS '94, l/94 Fairfax Va, USA (1994)
8. Bertino, E., Bettini, C., Samarati, P.: A Temporal Access Control Mechanism for Database Systems. IEEE Transactions on Knowledge and DataEngineering 8(1) (Feburary 1996)
9. Bertino, E., Bettini, C., Ferrari, E., Samarati, P.: An Access Control Model Supporting Periodicity Constraints and Temporal Reasoning. ACM Transactionon Database Systems 23(3) (September 1998)
10. Bertino, E., Bonatti, P.A, Ferrari, E.: TRBAC: A Temporal Role-based Access Control Model. ACM Transactionon on Information and System Security 4(3), 191–233 (2001)
11. Kargl, F., Maier, J., Weber, M.: Protecting Web Servers from Distributed Denial of Service Attacks. In: Proceedings of WWW '10, pp. 514-525 (2001)
12. Graham, G.S., Denning, P.J.: Protection - Principles and Practice. In: Proceedings of the AFIPS Srping Joint Computer Conference, vol. 40, pp. 417–429. AFIPS Press (May 16-18, 1972)
13. James, B.D., Joshi, E., Bertino, U., Latif, A., Ghafoo, A.: A Generalized Temporal Role-Based Access Control Model. IACM Transactionon on Knoledge and Data Engineering 17(1), 4–23 (2005)
14. Park, J., Zhang, X., Sandhu, R.: The Usage Control Model. In: ACM Transactions on Information and Systems Security, ACM Press, New York (Feburary 2004)
15. Park, J., Zhang, X., Sandhu, R.: Attribute Mutability in Usage Control. IFIP WG 11.3 (November 2004)

16. Allen, J.F.: Maintaining Knowledge about Temporal Intervals. Communications of ACM 26 (November 1983)
17. Lo, J.: Denial of Service or "Nuke" Attacks (March 12, 2005), http://www.irchelp.org/irchelp/nuke/
18. Doerr, M., Yiortsou, A.: Implementing a Temporal Datatype. Technical Report ICS-FORTH/TR-236 (November 1998)
19. Kudo, M., Hada, S.: XML Document Security based on Provisional Authorization. In: CCS'00, Athens, Greece, ACM Press, New York (2000)
20. Weaver, N.: Warhol Worms: The Potential for Very Fast Internet Plagues, http://www.cs.berkeley.edu/nweaver/warhol.html
21. Griffiths, G.S., Wade, B.W.: An authorization mechanism for a relational database system. ACM Transactions On Database Systems 1(3), 242–255 (1976)
22. Sandhu, R.: Access Control: The Neglected Frontier (Keynote Lecture). In: Australasian Conference on Information Security and Privacy (1996)
23. Sandhu, R.: Role Hierarchies and Constraints for Lattice-Based Access Controls. In: European Symposium on Research in Security and Privacy (1996)
24. Sandhu, R., Park, J.: Usage Control: A Vision for Next Generation Access Control. In: Models and Architectures for Computer Networks Security. The Second International Workshopon Mathematical Methods (2003)
25. Siewe, F., Cau, A., Zedan, H.: A Compositional Framework for Access Control Policies Enforcement. In: Proceeding of the ACM Workshop on Formal Methods in Security Engineering, ACM Press, New York (2003)
26. Jajodia, S., Samarati, P., Subrahmanian, V.S.: A Logical Language for Expressing Authorizations. In: IEEE Symposium On Research in Security and Privacy, Oakland, California (1997)
27. Zhang, X., Park, J., Parisi-Presicce, F., Sandhu, R.: A Logical Specification for Usage Control. In: 9th ACM Symposium on Access Control Models and Technologies (SACMAT), ACM Press, New York (June 2-4, 2004)
28. Zhang, X., Parisi-Presicce, F., Park, J., Sandhu, R.: Formal Model and Policy Specification of Usage Control. ACM Transactions on Information and System Security (TISSEC) 8(4), 351–387 (2005)

New Paradigm of Inference Control with Trusted Computing

Yanjiang Yang, Yingjiu Li, and Robert H. Deng

School of Information Systems, Singapore Management University
80 Stamford Road, Singapore 178902
{yjyang,yjli,robertdeng}@smu.edu.sg

Abstract. The database server is a crucial bottleneck in traditional inference control architecture, as it enforces highly computation-intensive auditing for all users who query the protected database. As a result, most auditing methods, though rigorously studied, can never be implemented in practice for protecting largescale real-world database systems. To shift this paradigm, we propose a new inference control architecture that will entrust inference control to each users platform, provided that the platform is equipped with trusted computing technology. The trusted computing technology is designed to attest the state of a users platform to the database server, so as to assure the server that inference control could be enforced as expected. A generic protocol is proposed to formalize the interactions between the users platform and database server. Any existing inference control technique can work with our protocol, for which the security properties are formally proven. Since each user's platform enforces inference control for its own queries, our solution avoids the bottleneck.

Keywords: Inference control, trusted computing, auditing, security protocol.

1 Introduction

Inference problem. The inference problem has been a long standing issue in database security that was first studied in statistical databases [13,2] and then extended to multilevel databases and general-purpose databases [21]. The inference problem can be referred to as the concern that one can infer (sensitive) information beyond one's privileges from the data to which one is granted access. The inference problem cannot be solved by traditional access control, as the disclosure of sensitive information is not derived from unauthorized accesses but from authorized ones. The existence of various inference vulnerabilities is due to the inevitable interconnections between sensitive data that are protected from and non-sensitive data that are provided in users' accesses.

Figure 1 shows a simple example that helps to illustrate the inference problem. The employee table contains age and salary information for a group of employees. To protect individuals' salary information, the following access rule is enforced: *while the database server can answer queries about sums of salaries over multiple employees, any query about a single employee's salary is illegitimate, and thus should be denied*. With this access control enforced, however, employee A's salary can still be easily derived

NAME	AGE	SALARY
A	28	2800
B	29	3100
C	30	3200
D	31	3600
E	32	3000
F	33	3200

Fig. 1. Employee table

from the following legitimate queries q_1 and q_2 provided that A is the only employee whose age is 28:

$$q_1 : \text{select sum(SALARY) from EMPLOYEE where AGE} \geq 28 \text{ and AGE} \leq 32$$
$$q_2 : \text{select sum(SALARY) from EMPLOYEE where AGE} \geq 29 \text{ and AGE} \leq 32$$

Inference control. Inference poses a serious threat to the confidentiality of database systems. Extensive research has been conducted on inference control to mitigate the threat. Inference control techniques can be classified into four categories [2]: conceptual-modeling, data perturbation, output perturbation, and query restriction. The *conceptual-modeling approach* (e.g., [8,15,17]) investigates the inference problem from a high level perspective, presenting frameworks for inference control. The proposed frameworks are sometimes too general for practical implementation. The *data perturbation approach* (e.g., [23,32,29,38,53,56,57]) typically replicates the original database and generates a perturbed database with noise for users to access. This approach suffers from a severe bias problem due to the noise that is added into data; therefore, it is not suitable for dynamic databases. In contrast, the *output perturbation approach* (e.g., [1,3,14,22] does not add noise, but performs certain manipulations over database queries such as rounding the query replies up or down. Though the output-perturbation based approach is immune to the bias problem, it may suffer from having null query sets, in which case useful information is disclosed. The last category is the *query restriction approach*, which can be further classified into five sub-categories: query-set-size control (e.g., [16,44]), query-set-overlap control (e.g., [18]), auditing (e.g., [6,9,24]), partitioning (e.g., [7,60,45]), and cell suppression (e.g., [10,36,41]). A comparison of these methods is given in [2] from various aspects including degree of security, query processing overhead, and suitability for dynamic databases.

Auditing. Auditing is an important query restriction-based approach. Traditional auditing works on the server side. The server keeps a log of all users' queries and, whenever a new query arrives on the server side, checks for possible inference vulnerabilities against the new query as well as the past queries asked by the same user. Since the control decision is made based upon a user's whole access history, auditing has the potential to achieve better security. Furthermore, auditing provides users with unperturbed query results as long as no inference vulnerability is detected. Due to these features, auditing has triggered intensive research in database security from the 1970s [43,24] through the 1980s [9,6,5,4] and 1990s [11,34,58] to the 21st century [59,27,55,31,33,54,30].

Unfortunately, auditing faces enormous difficulty in practical deployment, mainly due to the excessive computational overhead it requires to check for inference vulnerabilities from the accumulated query log. Audit Expert[1] [9] is a typical example. It was shown that it takes Audit Expert $\mathcal{O}(n^2)$ time to process a new SUM query [9], where n is the number of database entities or records, and $\mathcal{O}(mn^2)$ time to process a set of m queries. While this workload could be improved to some extent in certain specific situations (e.g., for range queries [6]), the auditing complexity is significantly higher in more general cases. Noting that Audit Expert protects only real-valued attributes from being inferred exactly, Kleinberg et al. [27] studied the auditing of boolean attributes and proved intractable results. Li et al. [31] concluded that the problem of protecting bounded real or integer attributes from being inferred within accurate-enough intervals has much higher complexities than that of Audit Expert.

The high complexity in auditing results in low system scalability. The database server can only afford a small number of users querying simultaneously. For this reason, auditing is not deemed to be a practical inference control method for real world database systems [2].

Inference control with trusted computing. To resolve the impracticality problem, we propose a new architecture for inference control (especially auditing) with trusted computing. The new architecture entrusts the enforcement of inference control to individual users' computer platforms. In this new architecture, the database server is responsible for the enforcement of traditional access control, while each user's platform is empowered to handle inference control based on their own query logs in a decentralized manner. Since the computation-intensive task of auditing is amortized to all users, the database server is no longer a bottleneck. As a result, our architecture can potentially be used for protecting large-scale database systems.

Since the inference control is enforced on the user side, it is crucial to ensure that the enforcement is conducted exactly as expected by the database server, without any interference or manipulation. This requires that each user's platform is in a trusted state when the inference control is enforced. A typical solution to attain this is to equip each user's machine with a TCG-compliant trusted platform module (TPM) [52] that establishes a hardware-based chain of trust from booting to OS loading to application execution. In our architecture, TPM is used to protect the execution environment of inference control and attest the trusted state of a user's platform to the remote database server when inference control is enforced. A generic protocol is proposed to formalize the interactions between the user's platform and the database server. Any existing inference control technique can work with our protocol, for which the security properties are formally proven.

Paper organization. The rest of this paper is organized as follows. In Section 2, we propose a new architecture for shifting the inference control paradigm. In Section 3, we present a protocol to enable inference control to be executed on the users' side using standard TPM commands. In Section 4, we discuss some extensions to our solution. Finally, Section 5 concludes this paper.

[1] Audit Expert is a classic auditing method. It maintains a binary matrix whose columns represent specific linear combinations of database entities (records) and whose rows represent user queries that have already been answered. Audit Expert transforms the matrix by elementary row operations to a standard form and concludes that exact inference exists if at least one row contains all zeros except in one column.

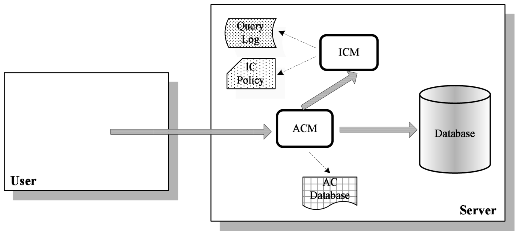
2 Architecture

Traditional architecture. The traditional architecture for inference control is illustrated in Figure 2(a), where both access control and inference control are enforced at the database server side. In this architecture, the access control module (ACM) implements access control functionality, while the inference control module (ICM) executes a designated inference control algorithm (e.g., Audit Expert) and acts as an extra line of defense in protecting the database. Upon receiving a new query from a user, ACM first decides whether the user is a legitimate user with respect to the queried data. This can be done by checking an access control database (AC database), which contains access control rules and policy. If the user is legitimate, the database server further checks with ICM to determine whether the query will lead to any information disclosure. ICM assesses the query against the inference control (IC) policy as well as the user's past queries (collected in the query log) by executing the designated inference control algorithm. The response to the query is returned to the user only if ACM decides that the user has the proper access right and if ICM concludes that no information disclosure would occur under the inference control policy. In this architecture, the IC policy is an essential component that stipulates what is necessary for the execution of the inference control algorithm, e.g., protection attributes, objectives, and constraints. The query log is maintained by the server, which accumulates all queries asked by each user.

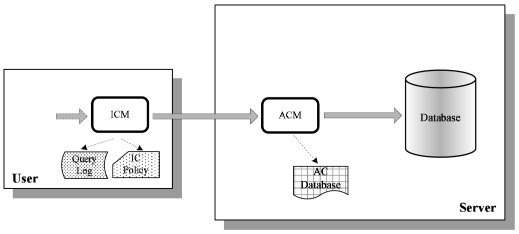
New architecture. Since the enforcement of inference control is computationally intensive, it may bottleneck the database server in the traditional architecture. To solve this problem, we propose a new architecture, shown in Figure 2(b), for inference control. The basic idea is to offload the inference control function to individual users. More specifically, ICM resides at the user side instead of on the server side. ICM maintains a query log by accumulating the queries issued by the user. To query the database, the user contacts ICM by issuing a query, then ICM checks with ACM at the server side to see whether the user has the right to access the data. ACM checks the user's request against the access rules and policy. If the user is granted access, ACM returns the query response, together with the IC policy[2], to ICM. Then, ICM executes the inference control algorithm by checking the query against its query log and IC policy. ICM releases the response to the user only if the query would lead to no information disclosure under the IC policy. In this architecture, ICM on the user side acts as an extension of the database server in inference control.

Role of trusted computing. A challenging issue in our new architecture is that the database server may lose its control over ICM, and that the user may compromise ICM so as to bypass inference control. To address this issue, certain kind of assurance must be given to the database server that ICM will be executed as expected, free of user's interference and manipulation. This kind of assurance is achieved by virtue of trusted computing. In Figure 3, a user's machine is equipped with a TCG-compliant TPM [52] and possibly other trusted hardware. A trusted platform can be built based on TPM at the hardware layer, as well as a secure kernel in the OS kernel space and ICM in the application space.

[2] The IC policy can be delivered to the user each time it is modified by the server; otherwise, it can be kept at the user's platform safely (protected by TPM).



(a) Traditional inference control architecture



(b) New inference control architecture

Fig. 2. Inference control architectures

The hardware, underpinning and cooperating with the secure kernel, provides necessary security functions to ICM, from basic cryptographic functions to sealed storage, platform attestation, and a protected running environment. TPM protects the integrity of the components in the platform, including the secure kernel and ICM, through its integrity measuring, storing, and reporting mechanisms. More importantly, the running state of the protected platform can be conveyed to the remote database server by virtue of the platform attestation mechanism of TPM, so that the server can decide whether the protected platform runs in a sufficiently trusted state. The protected platform running in a trusted state ensures that ICM performs inference control as expected, free of user's interference or manipulation. This platform architecture can be considered as an open system in the sense that the host accommodates both protected applications and unprotected applications.

The involvement of TPM in inference control can be considered yet another application of trusted computing [35]. Other security applications that have been rigorously studied in recent years include digital rights management [20], remote access control enforcement [39, 42], server-side user privacy protection [26], server privacy protection [48], secure auction [37], and integrity measurement [40], to name a few. The objective of these applications is to enable a server to extend its control over data distributed to client sides, or protect users' privacy on the server side, while our major concern is to securely decentralize the enforcement of inference control so as to resolve the efficiency and scalability problems inherent in inference control. For simplicity

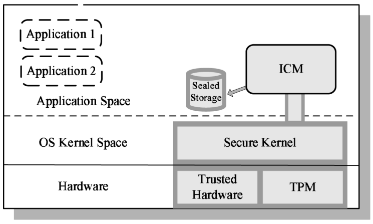


Fig. 3. TPM-enabled user host

reasons, we assume prior knowledge about TPM. Interested readers are referred to [35] for a brief review of TPM.

Interactions between ACM and ICM. In our new architecture, ACM enforces the access control mechanism over the database, and it represents the database server by interacting with all users. On the other hand, ICM is responsible for enforcing inference control according to the IC policy specified by the database server, and it also acts as an interface of the user side interacting with the database server. ICM is an application protected by TPM, which is inextricably bound to the user host.

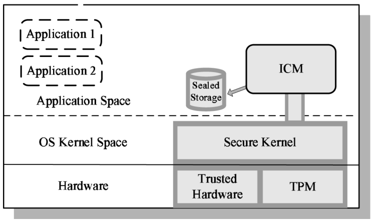


Fig. 4. Interactions between ACM and ICM

The interactions between ICM (having access to TPM) and ACM are illustrated in Figure 4. It is assumed that the user has certain identification information (e.g., user password) to identify itself to ACM. When ICM sends the user's identification information together with a user query to ACM (database server), ACM enforces access control and formulates a response to the user's query if the user query is authorized. Before ACM delivers the query's response, it first sends an attestation challenge to ICM. Based on the attestation response from ICM, ACM can decide whether the user's platform is in a trusted state. If so, it releases the query's response as well as the IC policy to ICM for the enforcement of inference control. A detailed protocol is given in the next section to formalize the interactions between ACM and ICM.

3 Protocol

In this section, we present a protocol for the interactions between ACM and ICM. The protocol enables ICM at the user side to enforce the inference control prescribed by the database server. The designing of the protocol assumes the use of version 1.2 TPM command set [51] and data structure [50].

3.1 Overview

Shifting inference control from the database server to users' hosts incurs new security threats that do not exist in the traditional architecture. We enumerate the new security threats and explain the basic ideas to mitigate these threats. The fundamental assumption is that the TPM is perfectly secure in the sense that the functions of TPM cannot be compromised. We note that there may exist some attacks that modify or crash the protected applications at user side after they have been attested by server. Attacks of these kinds are not specific to our system, but generic to all applications of trusted computing technology. A simple solution suggested by [39] is that the server frequently challenges the user's platform so as to detect and thwart these types of attacks.

Integrity of ICM. Since ICM resides in the user's host, a malicious user clearly has a motivation to alter the designated function of the protected platform, especially ICM, so as to bypass the inference control. Since TPM is inextricably bound to the user's host, we can use its integrity measuring, storage, and reporting mechanisms to detect any compromise of the integrity of the user's platform, including ICM.

Integrity of query log. Since the enforcement of inference control depends on the query log (which is maintained by ICM in the user's host), any unauthorized modification including deletion of the query log would render inference control baseless. To thwart this threat, a straightforward solution is to let ICM hold a secret key for either MACing or signing the query log, with the secret key stored in the sealed storage of TPM. However, this introduces an extra key for ICM to manage. We instead use a different method by associating the integrity digest of the query log with the key for protecting the confidentiality of query responses.

Authenticity of IC policy. The IC policy regulates how inference control is enforced. While in transit or in storage, the IC policy is subject to malicious alteration. It is extremely important to ensure that the IC policy enforced by ICM in the user's host is indeed dispatched by the database server and has not been tampered with. This is achieved by assuming that ACM holds a digital signature key pair (pk_{ACM}, sk_{ACM}). Before disseminating the IC policy to the user, ACM signs the IC policy so that ICM can check whether the policy has been compromised either in transit or in storage. This also enables ICM to verify the source of the IC policy.

Confidentiality of secrets maintained by ICM. In some cases, ICM needs to maintain some secret keys so as to protect the database server's data on the user side. To prevent malicious users from reading the secrets, ICM needs help from TPM to store these secrets in the sealed storage provided by TPM.

Confidentiality of query responses. Before ICM determines whether it is safe to release query responses, the user should be kept from reading the responses, whether they are in transit or in store. While in store, the query responses can be protected using secret keys maintained by ICM (as described above). To achieve the confidentiality of query responses in transit, a secure channel between ICM and ACM is established. More specifically, ICM asks TPM to generate an ephemeral asymmetric encryption key pair, where the public key is certified by TPM and the private key is stored in its protected storage. The public key can be used by ACM to encrypt the query responses, which will be sent to ICM. Upon receiving the encrypted message, ICM asks TPM for decryption operation in a secure software environment. Note that in this solution, TPM acts as a certifying party; there is no need to resort to external certification mechanisms.

As stated earlier, we novelly integrate the integrity protection of a query log into the confidentiality protection of query responses. This is explained as follows. When requesting that TPM perform the decryption operation, the invoking entity (i.e., ICM) is required to provide a piece of authorization data, which is normally derived from a password that is provided by the user who invokes ICM. In our solution, however, the piece of authorization data is derived by ICM not only from the user's password but also from a content digest of the query log. As a result, if the integrity of the query log is compromised, the authorization data will be refuted by TPM so that the private key cannot be accessed for the decryption operation. We must point out that this content digest is not intended to enhance the secrecy of the authorization data, which depends totally on the strength of the user's password.

Protected execution environment. A protected execution environment is needed for the running of ICM; otherwise, the OS kernel or other applications running in parallel on the user's host may access the code and data within the ICM application domain. Though a TPM-enabled platform can be configured as a restricted system (in which only a small set of protected applications such as ICM can run) or an open system but with all applications being protected by TPM, neither of the systems is practical. While the impracticality of the restricted system is obvious, it is challenging for TPM to perform platform attestation in an open system. The reason is that the attestation would involve a large set of application integrity metrics and that the database server must know in advance all the applications that run on each user's platform.

A more practical solution is that the user host remains open, but it is partitioned into a protected domain and an unprotected domain. The protected domain consists of a restricted set of protected applications such as ICM, while the unprotected domain includes other application softwares that do not need to be protected. Although the current TPM functionalities specified by the Trusted Computing Group (TCG) do not suffice to support this solution, more efforts have been made to establish the protected environment as desired. For example, the Intel's LaGrande Technology (LT) [28] incorporates an additional set of hardware and software components around the TCG-compliant TPM, which provides a protected execution environment that is sufficient for our solution. Without further complicating our presentation, it is reasonable to assume in our protocol that TPM (possibly together with other trusted hardware) enables ICM to run in isolation, free from interference by other applications running in parallel. Moreover, the application data that ICM uses in its execution domain will be automatically erased as long as ICM exits its execution.

3.2 Steps

We present our protocol in five steps, where the first four steps correspond to the four stages of interactions shown in Figure 4, and the last step represents the enforcement of inference control over the query response and the IC policy that ICM receives from ACM in the last stage of interaction. The following notation will be used in our presentation. Let $E_{pk}(.)$ and $D_{sk}(.)$ denote the encryption operation with public key pk and the decryption operation with private key sk, respectively. Let $enc(k, .)$ and $dec(k, .)$

denote encryption and decryption with symmetric key k, respectively. Let $S_{sk}(.)$ denote a message-aware digital signature scheme with private key sk. Let $SHA1(.)$ denote SHA-1 hash function. Let $A \rightarrow B : m$ represent A sending message m to B.

Step 1. ICM $\rightarrow$ ACM: id_U, q
To issue query q, the user invokes ICM to send user identification information id_U together with q to ACM on the database server side. Without specifying the composition of the identification information, we simply assume that id_U suffices to enable ACM to identify the user and enforce access control.
Step 2. Upon receiving a query request from the user, ACM checks whether the user has the requested right to access the data in the query; if so, ACM challenges the user's platform for remote attestation. This step consists of three sub-steps.

Step 2.1. ACM: $identify(id_U)$

ACM identifies the user by executing $identify(id_U)$, the deployed identification function.

Step 2.2. ACM: $ac(id_U)$

ACM executes the access control algorithm $ac(id_U)$ to determine whether the user has the permission to access the data in the query. If the user is not authorized, ACM aborts the protocol; otherwise, it continues with step 2.3.

Step 2.3. ACM $\rightarrow$ ICM: n_{ACM}

ACM generates a random nonce n_{ACM} and sends it to ICM. The nonce is used to thwart replay attacks in the following platform attestation.
Step 3. The platform attestation is performed in this step. Before the start of this step, ICM has in possession a public key pk_{ICM} generated by TPM in the last query session. This will be clear shortly (in steps 5.7 and 5.8)[3].

Step 3.1. ICM $\rightarrow$ TPM: TPM_CertifyKey

ICM first invokes a standard TPM command TPM_CertifyKey for TPM to certify pk_{ICM}. The TPM_CertifyKey command instructs TPM to generate a signature on a public key using its attestation identity key (AIK). The operation of key certification can be bound to a specific state of the underlying platform. The input parameters of TPM_CertifyKey include the key to be certified, externally supplied data of 20 bytes, and the Platform Configuration Registers (PCRs), whose contents are bound to the certification operation. The externally supplied data is calculated from $SHA1(n_{ACM})$, and the PCRs contain the integrity measurement metrics for the protected platform including the booting procedure, the OS, and ICM.

Step 3.2. TPM $\rightarrow$ ICM: TPM_Certify_Info, $\sigma_{TPM} = S_{sk_{TPM}}(SHA1(pk_{ICM}) || SHA1(n_{ACM}) || im)$

In response, TPM outputs a TPM_Certify_Info data structure, as well as a signature signed on the public key pk_{ICM}, the nonce n_{ACM}, and the integrity measurement metrics im of the platform. Here TPM_Certify_Info contains information regarding the usage of the public key pk_{ICM}, the PCRs involved in signing, and a digest of the public key. Note that sk_{TPM} (resp. pk_{TPM}) denotes the private (resp. public) AIK of TPM.

[3] In the case that the user queries the database server for the first time, there will be two extra sub-steps prior to step 3.1 that enable ICM to generate pk_{ICM}. The two extra sub-steps are the same as steps 5.7 and 5.8, with the only exception that the user's query log is empty at this point.

For the sake of simplicity, an atomic quantity im is used to represent the integrity measurement metrics of the protected platform. It is interesting to note that σ_{TPM} serves as not only certification of pk_{ICM}, but also platform integrity reporting of im.

Step 3.3 ICM $\rightarrow$ ACM: TPM_Key, TPM_Certify_Info, σ_{TPM}, TPM.AIK credential

In response to the attestation challenge, ICM sends TPM_Key, TPM_Certify_Info, σ_{TPM}, and the relevant TPM AIK credential to ACM on the server side. Here TPM_Key is a data structure that is generated in the last query session; it contains the public key pk_{ICM} and other related information, as will be explained in step 5.8.

Step 4. ACM verifies the attestation response and sends a query response as well as the IC policy to ICM for the enforcement of inference control.

Step 4.1. ACM: $verify(\sigma_{TPM})$

Upon receiving the attestation response, ACM first verifies the signature σ_{TPM} using public key pk_{TPM} and the corresponding certificate information.

Step 4.2. ACM: $validate(im)$

Then, ACM verifies whether im (contained in TPM_Certify_Info) represents a trusted state of the user's platform as expected. In particular, it verifies whether ICM is running as expected. We use an atomic function $validate(.)$ to denote this process.

Step 4.3. ACM $\rightarrow$ ICM: $\varepsilon_1 = E_{pk_{ICM}}(k), \varepsilon_2 = enc_k(d), \sigma_{ACM} = S_{sk_{ACM}}(\varepsilon_1 || \varepsilon_2 ||$ IC policy $|| q || pk_{ICM}), pk_{ACM}$

If step 4.1 or 4.2 fails, the protocol aborts. Otherwise, ACM generates a secret key k for symmetric key encryption. It encrypts k using the public key pk_{ICM}, yielding ε_1. Then, it encrypts the query response d using k, yielding ε_2. After formulating the IC policy that is to be enforced by the user, ACM signs the IC policy, ε_1, ε_2, query q, and pk_{ICM} using its private key, yielding digital signature σ_{ACM}. Finally, ACM sends ε_1, ε_2, σ_{ACM}, and pk_{ACM} (including this public key's certificate) to ICM.

Step 5. ICM enforces inference control over the query response and IC policy in a protected execution environment supported by TPM.

Step 5.1. ICM: $verify(\sigma_{ACM})$

Upon receiving the query response, ICM verifies the signature σ_{ACM}. If the signature is genuine, it proceeds to the next step.

Step 5.2. ICM $\rightarrow$ TPM: TPM_LoadKey2

ICM issues TPM command TPM_LoadKey2 to TPM so as to load the private key sk_{ICM} to TPM. The input parameters taken by TPM_LoadKey2 include a TPM_KEY structure and authorization data. The TPM_KEY structure specifies the clear public key pk_{ICM} and the wrapped private key sk_{ICM} (which can be unwrapped by TPM), as well as information on PCR values bound to the key pair. The authorization data is computed from the user's password and the digest of the query log: $SHA1$(password||digest-of-query-log). Please refer to steps 5.7 and 5.8 for the exact composition of TPM_KEY, why the authorization data is computed in such way, and how digest-of-query-log is obtained.

Step 5.3. TPM $\rightarrow$ ICM: $k = D_{sk_{ICM}}(\varepsilon_1)$

Once TPM decides that the protected user platform is in a trusted state, and that the authorization data matches that specified when the ICM key pair was generated (see step 5.7), TPM unwraps sk_{ICM}, uses it to decrypt ε_1, and returns k to ICM.

Step 5.4. ICM: $d = dec_k(\varepsilon_2)$

ICM decrypts ε_2 using key k to get the query response d.

Step 5.5. ICM: $infcon(q_d, Q,$ IC policy$)$

ICM enforces inference control based on q_d, Q and the IC policy, where q_d denotes the current query q as well as its response d, and Q denotes the set of past queries as well as their responses (obtained from the query log). For reasons of generality, we assume that the query responses are used in inference control, though they are not absolutely necessary for data independent algorithms such as Audit Expert. ICM reveals d to the user if $infcon(.)$ arbitrates that q is safe, leading to no information disclosure through inference; otherwise, ICM refuses to release d and proceeds to step 5.7.

Step 5.6. ICM: $Q = Q \cup \{q_d\}$

ICM updates the query log Q by adding q_d. Note that Q remains unchanged if q_d causes inference.

Step 5.7. ICM $\rightarrow$ TPM: TPM_CreatWrapKey

In this step, ICM invokes TPM command TPM_CreatWrapKey to instruct TPM to generate an asymmetric key pair (pk_{ICM}, sk_{ICM}) and to wrap the private key sk_{ICM}. The input parameters of this command include (i) the handle of a wrapping key that can perform key wrapping, (ii) the authorization data necessary to access the wrapping key, (iii) a set of PCRs whose contents are bound to the wrapping operation, and (iv) the information about the key to be generated (e.g., key length, key algorithm, key usage).

The piece of authorization data is a SHA-1 hash value (20 bytes) that is required for unwrapping the wrapped data. In our scenario, the piece of authorization data is derived from the user's password and the content digest of the user's query log; that is, $SHA1$(password||digest-of-query-log), where digest-of-query-log is obtained by applying a one way function to the whole set of the user's queries accumulated in the user's query log. If the user's query log Q is maliciously modified later, the authorization data calculated for unwrapping operation in the next query session will be refuted by TPM and, as a result, sk_{ICM} cannot be accessed by ICM.

The key pair generated by TPM_CreatWrapKey is bound to a state of the platform. The binding is achieved by specifying a set of PCRs whose contents are bound to the wrapping operation. In our case, the PCRs record the integrity measurement metrics of the protected platform. This binding ensures that (sk_{ICM}) cannot be unwrapped unless the user's platform is in a trusted state.

Step 5.8. TPM $\rightarrow$ ICM: TPM_Key

Finally, TPM returns to ICM a TPM_Key data structure, which contains public key pk_{ICM}, and the corresponding private key sk_{ICM} encrypted by a wrapping key. TPM_Key also contains a field TPM_Auth_Data_Usage, which can take one of the following three values:

(i) TPM_Auth_Never, (ii) TPM_Auth_Always, and (iii) TPM_Auth_Priv_Use_Only

The first case allows the invoking party to load the private key without submission of any authorization data, while the second and third cases associate authorization data with the public/private key pair and the private key only, respectively. In our case, it suffices to indicate TPM to set TPM_Auth_Data_Usage to TPM_Auth_Priv_Use_Only.

3.3 Security

Given that the security services provided by TPM, the protected execution environment on top of TPM, and the cryptographic primitives we employed are secure (in a sense that these services are not compromised), it does not seem difficult to verify that our protocol meets the security requirements as posed by the security threats listed in Section 3.1. While mitigating these threats is a focus in our protocol design, the security of our protocol demands a formal analysis.

Under the assumption that the underlying TPM is perfect, our protocol can essentially be considered as an authentication protocol between ICM and ACM, aiming to satisfy the requirement that *ACM validates the security state of ICM before sending out any query response*. The security of our protocol can be formally proven using the rank function [46], a specialized theorem-proving method for establishing the correctness of authentication protocols based on communicating sequential processes (CSP) [47]. This will eliminate typical attacks such as replay attacks and masquerade attacks that are targeted at our authentication protocol.

To perform the proof, the rank function approach requires modeling the following three components: (1) the protocol, (2) the environment (attacker), and (3) the security requirements on the protocol. In particular, first, the protocol is captured as a CSP process in terms of the behavior of each system party; second, the environment is also described as a CSP process. It is considered to be an unreliable medium that can lose, reorder, and duplicate messages. The particular behavior captured within the medium is precisely the behavior that the protocol is designed to overcome; third, the security requirements on the protocol are expressed as **sat** specifications on the observable behaviors of the overall system. When these components are modeled, one can use well-established proof techniques to verify whether the protocol satisfies its security requirements. For limit of space, we omit the detailed proof.

4 Extensions

Defending against collusion attack. A typical attack against inference control systems is *collusion attack*. A collusion attack involves several users forming a *collusion group* and combining their query logs so as to infer some sensitive information that cannot be derived from any individual query log. The collusion attack is inherently difficult to mitigate, and presents as a serious inhibitor in the practical use of inference control [2]. We must point out that there seems no technique can prevent a user from purposely recording his/her queries (as well as query responses) and using them in collusion with other users. What we can achieve is to restrict malevolent users from directly using the query logs that are maintained by ICMs for collusion. This requires the *confidentiality of query log* to be maintained against any programs other than ICM. To attain this requirement, the query log can be encrypted by ICM using a secret key, which can be stored in and retrieved from the sealed storage of TPM by ICM only (using TPM_Seal and TPM_Unseal commands). Note that in this case, the authorization data for unwrapping operations must be changed to $SHA1$(password||digest of encrypted query log) in our protocol.

In certain cases, the database server may be able to "blacklist" some collusion groups of suspicious users who may collude using certain out-of-band information (e.g., the users from the same network domain). To further mitigate the collusion attack, inference control should be enforced based on queries from all users in a collusion group rather than from each individual user. While such control can be easily enforced with a central query log in the traditional architecture, it is not as easy to combine many users' query logs in our new architecture. A possible solution is to extend the new architecture such that the database server manages a central query log as in the traditional architecture. When any user in a collusion group issues a query, the server sends to the user's ICM the queries from all other users in the same collusion group. ICM can then enforce inference control based on the combination of its own queries and the queries it receives. To maintain the confidentiality of the query log, the queries should be sent from ACM to ICM in an encrypted form, which can be as easily done in our protocol as encrypting the query responses. Upon receiving them, ICM may keep these queries in its execution domain and delete them after use. Alternatively, ICM can add these queries to its query log such that the query log contains queries from a collusion group instead of from an individual user (this would substantially decrease the number of queries sent each time by the database server).

User using multiple hosts. Our protocol is essentially designed for the scenario in which a user is bound to a single host. This can be seen that each user's query log is maintained at the user's host and the user's host accumulates the queries issued from that particular host. If a user is allowed to use multiple hosts to query the database, the query logs maintained at different hosts may not be readily available to the current host where inference control is enforced. A convenient solution is to let the database server maintain a central query log that collects user queries at the discretion of user identity in conjunction with host identity[4]. When a user issues some queries from a host, the queries previously issued by the user from all other hosts are passed by ACM to the current user host for the enforcement of inference control.

Database update. Database update (e.g., deletion of some records) may necessitate updating the user's query log on the user's side. To facilitate updating, the database server may manage a central query log as discussed above. In case of database update, the database server can determine the set of queries that are affected by the update process. When a user queries the database, the database server first checks for the affected queries that belong to the user; it then informs the user to update its query log so that the inference control will be enforced upon the latest query log.

5 Conclusions

This paper proposed a new inference control architecture and an inference control protocol upon it. While traditional inference control is enforced by a database server for all its users, our solution entrusts each user's host to enforce inference control for itself, provided that the user's host is equipped with trusted computing technology. By decentralizing the highly computation-intensive task of inference control, our solution enjoys

[4] The TPM bound to a user's host can be used to unambiguously identify the host.

much better system scalability, and is thus suitable for supporting a large number of users in real world database systems. In comparison, the traditional inference control configuration can only support a small number of users due to the bottleneck of enforcing inference control for all users on the server side. In this sense, our solution removes the crucial impediment in traditional inference control configuration and identifies a new paradigm for the practical implementation of inference control.

Our solution relies on the adoption of widely available trusted computing technology, which is envisioned to be ubiquitous in several years. This trusted computing technology is utilized by the database server to attest users' platforms so that the inference control can be enforced on the user side as expected by the database server. The security properties of our solution are formally proven with the rank function approach. Our solution can work with any existing inference control technique; even a hybrid system of mixing our solution (for some users whose platforms are TPM equipped) with traditional inference control (for those users who may not implement trusted computing) can be easily set up. An interesting future direction is to implement our solution with various existing inference control techniques in some real world settings.

References

1. Achugbue, J.O., Chin, F.Y.: The Effectiveness of Output Modification by Rounding for Protection of Statistical Databases. INFOR 17(3), 209–218 (1979)
2. Adam, N.R., Wortmann, J.C.: Security-Control Methods for Statistical Databases: A Comparative Study. ACM Computing Surveys 21(4), 516–556 (1989)
3. Beck, L.L.: A Security Mechanism for Statistical Databases. ACM Trans. Database Systems 5(3), 316–338 (1980)
4. Chen, M., McNamee, L., Melkanoff, M.A.: A Model of Summary Data and Its Applications to Statistical Databases. In: Rafanelli, M., Svensson, P., Klensin, J.C. (eds.) Statistical and Scientific Database Management. LNCS, vol. 339, pp. 354–372. Springer, Heidelberg (1989)
5. Chin, F.Y.: Security Problems on Inference Control for SUM, MAX, and MIN queries. J. ACM 33, 451–464 (1986)
6. Chin, F.Y., Kossowski, P., Loh, S.C.: Efficient Inference Control for Range Sum Queries. Theor. Comput. Sci. 32, 77–86 (1984)
7. Chin, F.Y., Özsoyoglu, G.: Security in Partitioned Dynamic Statistical Databases. In: Proc. IEEE COMPSAC, pp. 594–601. IEEE Computer Society Press, Los Alamitos (1979)
8. Chin, F.Y., Özsoyoglu, G.: Statistical Database Design. ACM Trans. Dababase Systems 6(1), 113–139 (1981)
9. Chin, F.Y., Özsoyoglu, G.: Auditing and Inference Control in Statistical Databases. IEEE Trans. Softw. Eng. 6, 574–582 (1982)
10. Cox, L.H.: Suppression Methodology and Statistical Disclosure Control. J. Am. Stat. Assoc. 75(370), 377–385 (1980)
11. Cox, L.H., Zayatz, L.V.: An Agenda for Research on Statistical Disclosure Limitation. J. Official Statistics 75, 205–220 (1995)
12. Delicata, R.: An Analysis of Two Protocols for Conditional Access in Mobile Systems, Technical Report CS-04-13, Department of Computing, University of Surrey (2005)
13. Denning, D.E.: Cryptography and Data Security. Addison-Wesley, Reading (1982)
14. Denning, D.E.: Secure Statistical Databases with Random Sample Queries. ACM Trans. Database Systems 5(3), 88–102 (1980)

15. Denning, D.E.: A Security Model for the Statistical Database Problem. In: Proc. 2nd International Workshop on Management, pp. 1–16 (1983)
16. Denning, D.E., Denning, P.J., Schwartz, M.D.: The Tracker: A threat to Statistical Database Security. ACM Trans. Database Systems 4(1), 76–96 (1979)
17. Denning, D.E., Schlörer, J.: Inference Control for Statistical Databases. Computer 16(7), 69–82 (1983)
18. Dobkin, D., Jones, A.K., Lipton, R.J.: Secure Databases: Protection Against User Influence. ACM Trans. Database Systems 4(1), 97–106 (1979)
19. Dolve, D., Yao, A.C.: On the Security of Public Key Protocols. IEEE Transactions on Information Technology 29(2), 198–208 (1983)
20. Erickson, J.S.: Fair use, DRM, and trusted computing. Communications of ACM 46(4), 34–39 (2003)
21. Farkas, C., Jajodia, S.: The Inference Problem: A Survey. SIGKDD Explorations 4(2), 6–11 (2002)
22. Fellegi, I.P., Phillips, J.L.: Statistical Confidentiality: Some Theory and Applications to Data Dissemination. Ann. Ec. Soc. Meas. 3(2), 399–409 (1974)
23. Greenberg, B.G., Abernathy, J.R., Horvitz, D.G.: Application of Randomized Response Technique in Obtaining Quantitative Data. In: Proc. Social Statistics Section, America, Statistical Association, pp. 40-43 (1969)
24. Hoffman, L.J.: Modern Methods for Computer Security and Privacy. Prentice-Hall, Englewood Cliffs (1977)
25. Hui, M.L., Lowe, G.: Safe Simplifying Transformations for Security Protocols. In: Proc. 12th Computer Security Foundations Workshop, pp. 32–43 (1999)
26. Iliev, A., Smith, S.W.: Protecting User Privacy via Trusted Computing at the Server. IEEE Security and Privacy 3(2), 20–28 (2005)
27. Kleinberg, J., Papadimitriou, C., Raghavan, P.: Auditing Boolean Attributes. In: Proc. 9th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 86–91. ACM Press, New York (2000)
28. LaGrande technology architecture: Intel Developer Forum (2003)
29. Lefons, D., Silvestri, A., Tangorra, F.: An Analytic Approach to Statistical Databases. In: Proc. 9th Very Large Databases, pp. 260–273 (1983)
30. Li, Y., Lu, H., Deng, R.H.: Practical Inference Control for Data. In: Proc. IEEE Symposium on Security and Privacy, pp. 115–120. IEEE Computer Society Press, Los Alamitos (2006)
31. Li, Y., Wang, L., Wang, X.S., Jajodia, S.: Auditing Interval-based Inference. In: Proc. 14th Conference on Advanced Information Systems Engineering, pp. 553–567 (2002)
32. Liew, C.K., Choi, W.J., Liew, C.J.: A Data Distortion by Probability Distribution. ACM Trans. Database Systems 10(3), 395–411 (1985)
33. Malvestuto, F.M., Mezzini, M.: Auditing Sum-Queries. In: Proc. International Conference on Database Theory, pp. 504–509 (2003)
34. Malvestuto, F.M., Moscarini, M.: An Audit Expert for Large Statistical Databases, Statistical Data Protection, EUROSTAT, pp. 29-43 (1999)
35. Mitchell, C.: Trusted Computing, The Institution of Electrical Engineers, London, UK (2005)
36. Özsoyoglu, G., Chung, J.: Information Loss in the Lattice Model of Summary Tables Due To Suppression. In: Proc. IEEE Symposium on Security and Privacy, pp. 75–83. IEEE Computer Society Press, Los Alamitos (1986)
37. Perrig, A., Smith, S.W., Song, D., Tygar, J.D.: SAM: A Flexible and Secure Auction Architecture using Tusted Hardware. eJETA.org: The Electronic Journal for E-Commerce Tools and Applications 1(1) (2002)
38. Reiss, J.P.: Practical Data Swapping: The First Step. In: Proc. IEEE Symposium on Security and Privacy, pp. 36–44. IEEE Computer Society Press, Los Alamitos (1980)

39. Sailer, R., Jaeger, T., Zhang, X., van Doorn, L.: Attestation-Based Policy Enforcement for Remote Access. In: Proc. ACM Conference on Computer and Communications Security, pp. 308–317. ACM Press, New York (2004)
40. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and Implementation of a TCG-based Integrity Measurement Architecture. In: USENIX. USENIX Security Symposium, pp. 223–238 (2004)
41. Sande, G.: Automated Cell Supperssion to Reserve Confidentiality of Business Statistics. In: Proc. 2nd Workshop on Statistical Database Management, pp. 346–353 (1983)
42. Sandhu, R., Zhang, X.: Peer-to-Peer Access Control Architecture Using Trusted Computing Technology. In: Proc. ACM Symposium on Access Control Models and Technologies, pp. 147–158. ACM Press, New York (2005)
43. Schlörer, J.: Confidentiality of Statistical Records: A Threat Monitoring Scheme of On-line Dialogue. Methods Inform. Med. 15(1), 36–42 (1976)
44. Schlörer, J.: Disclosure from Statistical Databases: Quantitative Aspects of Trackers. ACM Trans. Database Systems 5(4), 467–492 (1980)
45. Schlörer, J.: Information Loss in Partitioned Statistical Databases. Comput. J. 26(3), 218–223 (1983)
46. Schneider, S.: Verifying Authentication Protocols with CSP. In: Proc. 10th Computer Security Foundation Workshop, pp. 3–17 (1997)
47. Schneider, S.: Concurrent and Real-time Systems: the CSP Approach. Addison-Wesley, Reading (1999)
48. Smith, S.W., Safford, D.: Practical Server Privacy Using Secure Coprocessors. IBM Systems Journal (special issue on End-to-End Security) 40, 683–695 (2001)
49. TCG. TPM Main: Part 1 Design Principles, TCG Specification Ver. 1.2, Revision 62 (2003), http://www.trustedcomputinggroup.org
50. TCG. TPM Main: Part 2 TPM Data Structure, TCG Specification Ver. 1.2, Revision 62 (2003), http://www.trustedcomputinggroup.org
51. TCG. TPM Main: Part 3 Commands, TCG Specification Ver. 1.2, Revision 62 (2003), http://www.trustedcomputinggroup.org
52. Trusted Computing Group (2006), http://www.trustedcomputinggroup.org
53. Traub, J.F., Yemini, Y., Wozniakowski, H.: The Statistical Security of A Statistical Database. ACM Trans. Database Systems 9(4), 672–679 (1984)
54. Wang, L., Li, Y., Wijesekera, D., Jajodia, S.: Precisely Answering Multi-dimensional Range Queries without Privacy Breaches. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, pp. 100–115. Springer, Heidelberg (2003)
55. Wang, L., Wijesekera, D., Jajodia, S.: Cardinality-based Inference Control in Sum-only Data Cubes. In: Gollmann, D., Karjoth, G., Waidner, M. (eds.) ESORICS 2002. LNCS, vol. 2502, pp. 55–71. Springer, Heidelberg (2002)
56. Warner, S.L.: Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. J. Am. Stat. Asso. 60(309), 63–69 (1965)
57. Warner, S.L.: The Linear Randomized Response Model. J. Am. Stat. Asso. 66(336), 884–888 (1971)
58. Willenborg, L., Waal, T.: Statistical Discolure Control in Practice. Lecture Notes in Statistics, vol. 111. Springer, Heidelberg (1996)
59. Willenborg, L., Waal, T.: Elements of Statistical Discolure. Lecture Notes in Statistics, vol. 155. Springer, Heidelberg (2000)
60. Yu, C.T., Chin, F.Y.: A Study on the Protection of Statistical Databases. In: Proc. ACM SIGMOD, pp. 169–181. ACM Press, New York (1977)

Security Patterns for Physical Access Control Systems

Eduardo B. Fernandez, Jose Ballesteros, Ana C. Desouza-Doucet, and Maria M. Larrondo-Petrie

Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, Florida 33431, USA
ed@cse.fau.edu, jballes2@fau.edu, adoucet@bluefrogsolutions.com, maria@cse.fau.edu

Abstract. Physical security has received increased attention after 9/11. However, access control to physical units has not been explored much. On the other hand, there is a rich literature on access control to information. These two areas appear converging as seen by recent products and studies. However, use of different notations and implementation details make this convergence harder. We need to try to take this convergence at a more abstract level first. Although introduced about 10 years ago, security patterns have recently become accepted by industry and two books on this topic have appeared recently. Security patterns for information access control have appeared but now we extend this concept to access for physical units. The unification of information and physical access control is just beginning but the strong requirements of infrastructure protection will make this convergence to happen rapidly. Examining existing systems, industry standards and government regulations, we describe, in the form of patterns, the core set of features a physical access control system should have. The paper illustrates the structure and use of these patterns.

Keywords: access control, intelligent buildings, physical access control, security, software patterns.

1 Introduction

Homeland security has brought an added interest in control of access to buildings and other physical structures. The need to protect assets in buildings and to control access to restricted areas such as airports, naval ports, government agencies, and nuclear plants to name a few, created a great business opportunity for the physical access control industry and a good amount of interest in the research community. One of the results of this interest was the recognition that access control to information and access control to physical locations have many common aspects. The most basic model of access control uses a tuple (s,o,t), subject, object, access type. If we interpret s as a person (instead of an acting executing entity), o as a physical structure (instead of a computational resource), and t as a physical access type (instead of resource access), we can make an analogy where we can apply known results or approaches from information access control. The unification of information and physical access control is just beginning but the strong requirements for infrastructure protection will make this convergence to happen rapidly. Another issue is the fact that there are standard network protocols for building automation, e.g. BACnet [1], which are totally different of the protocols used for manufacturing automation, e.g. DNP3 [2]. Both types of protocols define security standards, which means that a building intended for manufacturing would have two sets of incompatible security standards. We need some way to abstract the security requirements of the complete system without regard to specific standard details.

One way to achieve this unification is using a conceptual abstraction for the definition of security requirements; we consider here the use of analysis and security patterns for this purpose. A pattern is an encapsulated solution to a recurrent problem in a given context [3][4]. In particular, a security pattern defines a solution to a security problem [5]. In general, the use of patterns has been increasing in industry because of their potential to improve software quality. Although introduced about 10 years ago, security patterns have only recently become accepted by industry and Microsoft, IBM, and Sun have web pages on this topic. Also, two books have appeared recently [5][6]. We have presented several security patterns for access control to information, e.g. [7][8], and now we extend this concept to the access of physical units. Standards and products that deal with physical units use a set of common concepts that may appear different due to a different notation; patterns make this commonality apparent. Examining existing systems, industry standards, and government regulations, we describe, in the form of patterns, the relationship and definition of a core set of features a physical access control system should have. From these patterns, it is possible to define more specific patterns that can be used to build systems in a given protocol or to define new protocols. These patterns can also be combined to make up complex systems.

We present here patterns for access control in physical units. While we cannot be complete, we show three of them to illustrate their structure and possibilities:

– Alarm Monitoring. Defines a way to raise events in the system that might require special attention, like the tampering of a door.
– Relays. Defines the interactions with electronically controlled switches.
– Access Control to Physical Structures. Applies authentication and authorization (RBAC) to the control of access to physical units including alarm monitoring, relays, and time schedules that can control when things will happen.

The pattern diagram of Figure 1 shows how these patterns relate to each other and to related existing patterns. The patterns not explicitly described here include:

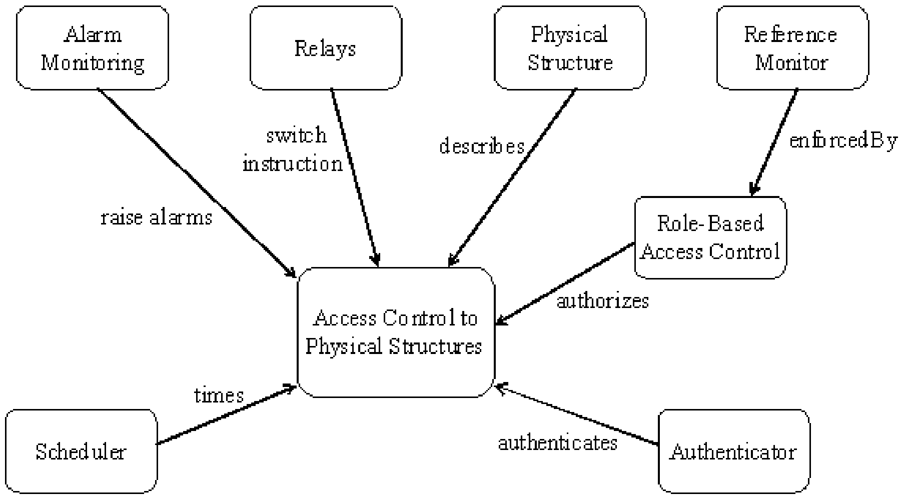


Fig. 1. Pattern diagram of physical access control patterns

– Physical Structure [9]. Defines the structure and use of physical sites such as buildings, parking lots, and similar, as well as their divisions and compartments.
– Scheduler. Provides timing information to control access.
– Role-Based Access Control [7]. Describes the standard RBAC model used here to describe authorization to access a physical unit.
– Reference Monitor [5]. Enforces authorizations when a process requests access to an object.
– Authenticator [10]. Verifies that a subject is who it says it is.

Alarm Monitoring, Relays, Scheduler, and Physical Structure are composed to form the Access Control to Physical Structures pattern. RBAC defines the type of authorization rules used in the system, while the reference monitor indicates an abstract pattern representing enforcement mechanisms. Authenticator is an abstract pattern defining the requirements for authentication. We present these patterns following the POSA template [3]. This template intends to provide enough detail to be a guideline for a designer building a system that requires this pattern. The set of patterns can also be used to define precise requirements and to evaluate existing systems. The solutions are described by UML diagrams, which although not strictly formal, are precise and unambiguous (UML has a well-defined syntactic metamodel). Because each pattern must be reusable on its own, there is some amount of redundancy between patterns. Each pattern is relatively simple but it can be combined with others to build complex systems.

Section 2 describes some background. Section 3 presents first two patterns that complement physical access (Alarm Monitoring and Relays). These patterns are then combined with other patterns to define a composite pattern that describes the necessary elements of a physical access control system. Section 4 includes a short discussion and comparison to other approaches. We end with some conclusions.

2 Background

Physical access control systems are widely used today and they can be implemented with a wide range of technologies. The basic idea is that something controls access to someplace; it could as simple as a key lock and as complex as a face recognition device. Moreover, there are ways to detect and inform when someone violates the access rules or tries to force the system. This simple definition leaves room for a great number of features and combinations in software and hardware that vary from product to product. To understand access control, we must understand the language of the industry. Terms like the ones discussed in the patterns presented here are commonly used when discussing access control systems. Other terms include:

- Access control panels. Serves as an interface to the readers and door locks. Wiring in a network interconnects most of these panels.
- Electric door locks. Keep the doors locked and secure, and release the door when a valid credential is used.
- Shunting of alarm devices. Means to bypass or ignore an alarm for a specified period of time.
- Anti-pass back. Used to prevent tailgating (when one user enters with a valid credential, and several people enter without using theirs).

As indicated earlier, a pattern describes a solution to a specific problem in a given context. Patterns are abstractions of real systems, emphasizing best practices and fundamental features, we found these patterns by analyzing real systems or standards. A security pattern describes an abstraction of a security mechanism able to avoid or mitigate some threats [5][6]. In addition to the two mentioned books about security patterns, a variety of security patterns have appeared in the literature [11]. It is common practice to describe the solution encapsulated in a pattern using UML (Unified Modeling Language) diagrams. UML is a standard for software development and its models are graphic and intuitive and can be conveniently converted into code. In addition, as indicated earlier, UML is a semi-formal language that provides a good amount of precision. There have been attempts to further formalize patterns but their importance comes not from an ability to prove security properties of the system but because of the fact that they are known to be good practices, based on experience. In addition, since each pattern is rather simple, they can be used by practitioners; many software developers know UML but are not able to use formal methods.

Patterns are described using some type of template, consisting of a specific set of sections with predefined meanings. We use here the so-called POSA template [Bus96] which includes the following sections:

- A *Name*, that should be unique and precise.
- A thumbnail *summary* of the pattern (what problem does it solve?)
- An *example* of a situation where a solution is needed.
- The *context* where the pattern is valid or applicable.
- The general *problem* solved by the pattern.
- A *solution* section, describing the idea of the solution. This includes two subsections, a *Structure* section describing a class diagram of the solution, and a *Dynamics* section describing some typical use case sequences.
- The *Implementation* section provides hints in using the pattern. The example resolved shows how the pattern can solve the problem of the example presented earlier.
- The *Known Uses* section indicates some real uses of the pattern.
- The *consequences* indicate the advantages and disadvantages of using this solution.
- The section on *related patterns* enumerate patterns which solve similar problems or are complementary to this pattern.

3 Patterns

3.1 Alarm Monitoring

Defines a way to raise events in the system that might require special attention, such as someone tampering with a door.

Example. Building management wants to raise alarm events when someone opens a door without using the right credentials or when someone tries to use a card that was reported as lost.

Context. Physical environment with an access control system where we want to be able to raise filtered alarm events and we want to differentiate between alarms that are generated because of a physical violation of the system and alarms generated because of a system rule violation.

Problem. In an Access control system, there are two types of alarms, physical and logical. Many times we might want inform more than one subsystem that a change of state happened in order to generate different types of configurable actions that can vary. Physical alarm inputs are used to monitor various devices. These alarms can be shunted. Logical alarms are used to monitor various system rules. For example, a system may generate an alarm after three invalid tries to get access with the same credentials.

The following forces will affect any solution:

- There are two types of alarms, physical and logical.
- Alarms can be ignored.
- Logical alarms have two possible states, reset and alarm.

– Physical alarms have four possible states: reset, alarm, cut or short. The last two states are known as trouble states, caused by faulty wiring or tampering.
– We may need to know what alarms have been set and when they were reset.
– We need a way to inform interested parties about a change of a state of an alarm.

Solution. Have an alarm entity that describes the general concepts of an alarm and use generalization to separate logical alarms from physical alarms and their particular characteristics. Add information about the time the alarm occurred. Use the Observer Pattern [4] to advise any interested party of a change of state in an Alarm.

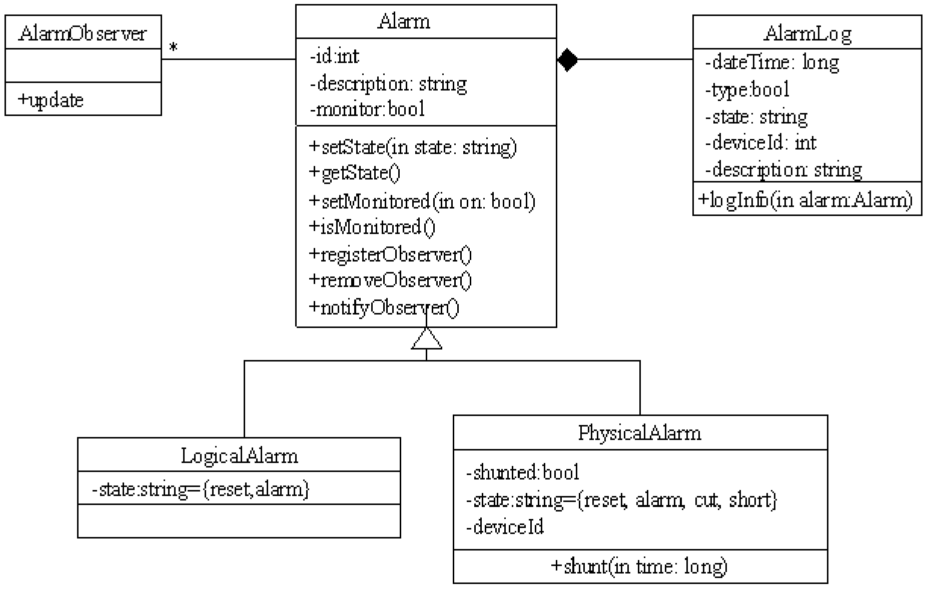


Fig. 2. Class Diagram for Alarm Monitoring

Structure Figure 2 shows a UML class diagram for the Alarm Monitoring pattern. Abstract class Alarm indicates a general class for any type of alarm. The PhysicalAlarm class and the LogicalAlarm class inherit the Alarm class. These classes allow for alarms to be controlled and monitored. By implementing the AlarmObserver interface any class (not shown here) interested on this alarm can be added to the list of observers for the alarm and will be advised when a change of state happens. Moreover, we log every alarm activity with a time stamp.

*Dynamics***.** Figure 3 shows the main success scenario for the use case "activate an alarm". The request to set the alarm active is sent by an external actor that detected the need to raise the alarm. The change of state is logged and only if the

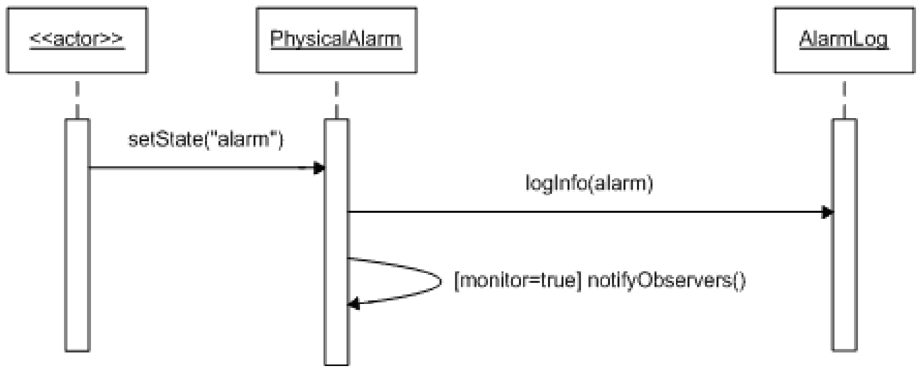


Fig. 3. Sequence Diagram for Activating an Alarm

alarm is being monitored interested parties are advised of the change, otherwise the alarm is ignored.

Implementation. There are many ways to create alarms in a physical access control system. For example, when a card reader is installed on a door, an alarm contact is usually installed as well. The alarm point is used to monitor whether the door was forced or held for too long after a valid access was granted. Logical Alarms can be generated for maximum tries with invalid credential, invalid credential and communication errors. The call to change the state of an alarm could in turn generate other actions when observers are notified, like displaying a message, activating a siren, etc.

Example Resolved. Every time someone opens a door without proper permission an alarm can be created and if a person uses a lost badge the system can generate a logical alarm.

Known Uses. Many commercial Access Control systems have the concept of logical and physical alarms that generate some actions when the states are changed.

Consequences. Advantages include: We can only pay attention to the alarms we are interested in.

- Every alarm change of state is logged, and interested parties are advised.
- This model provides the basic structure for supporting alarms in an access control system.
- We make a distinction between logical and physical alarms, which supports the creation of any alarm independently of the existing hardware.

A disadvantage is: The pattern may create overhead for systems that only care about logging the alarm.

Related Patterns. This pattern is based on the Observer Pattern [4]. The Access Control for Physical Structures [9] complements this pattern by adding the concept of Zones that control Alarms.

3.2 Relays

This pattern defines the interactions with electronically controlled switches.

Example. Building management wants to be able to open the main gate for visitors that do not have credentials. Moreover, doors should be opened when someone presents valid credentials.

Context. Physical environment with an access control system where we want to be able turn on/off devices; and lock/unlock doors.

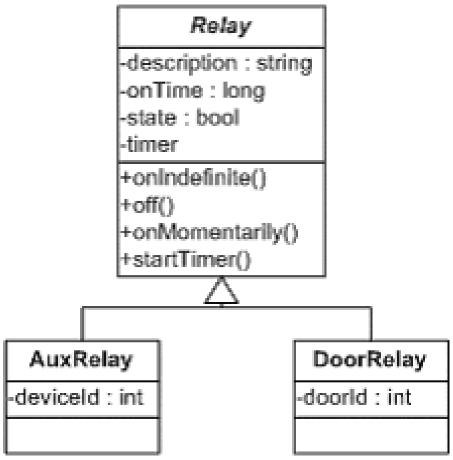


Fig. 4. Class Diagram for Relays

Problem. A relay is an electronically controlled switch. Similar to a light switch on the wall being used to turn on or off a light, a relay can be used to turn on or off other devices. Relays are used to activate the electric door lock, or to activate a variety of other items such as: bells, sirens, turn lights on and off, trip a digital dialer and many other uses. Typically, one relay is used to control the electric strike of doors. The others, usually called auxiliary relays, can be used as needed. When a relay is activated or deactivated, the device wired to it is turned on or off.

The following forces will affect the solution:

– We need to distinguish between door relays and auxiliary relays.
– Relays can be activated and deactivated.

– Relays can be activated indefinitely or for a defined period of time.
– Relays have two possible states: on and off.

Solution. Have a relay entity that describes the general concepts of a relay and use generalization to separate door relays from auxiliary relays and their particular characteristics.

*Structure***.** Figure 4 shows a UML class diagram for the Relays pattern. Abstract class Relay indicates a general class for any type of relay. The DoorRelay class and the AuxRelay class inherit the Relay class. These classes allow for relays to be controlled.

*Dynamics***.** The sequence diagram is trivial. The request to set the relay active is sent by an external actor. The relay is activated for the previously defined "on time." If the relay was already active the timer is restarted. Once the timer expires the relay is turned off.

Implementation Relays could be activated or deactivated by a variety of events. An alarm input, a valid credential, an egress button being pushed, or a time event, can all activate a relay.

Example Resolved. Doors that have relays defined can be opened when a valid credential is presented. Also the main gate can be assigned an auxiliary relay that can be activated at will.

Known Uses. Many commercial Access Control systems have the concept of relay management and distinction between door and aux relays.

Consequences. Advantages include:

– We can distinguish between auxiliary and door relays.
– We can activate relays for a predefined amount of time.
– This model provides the basic structure for supporting relays in an access control system.

A disadvantage is:

– We might want the same kind of functionality for devices that are not exactly door or aux relays.

Related Patterns. The Access Control for Physical Structures [9] complements this pattern by adding the concept of Zones that control Relays.

3.3 Access Control to Physical Structures

Applies authentication and authorization to the control of access to physical units including alarm monitoring, relays, and time schedules that can control when things will happen.

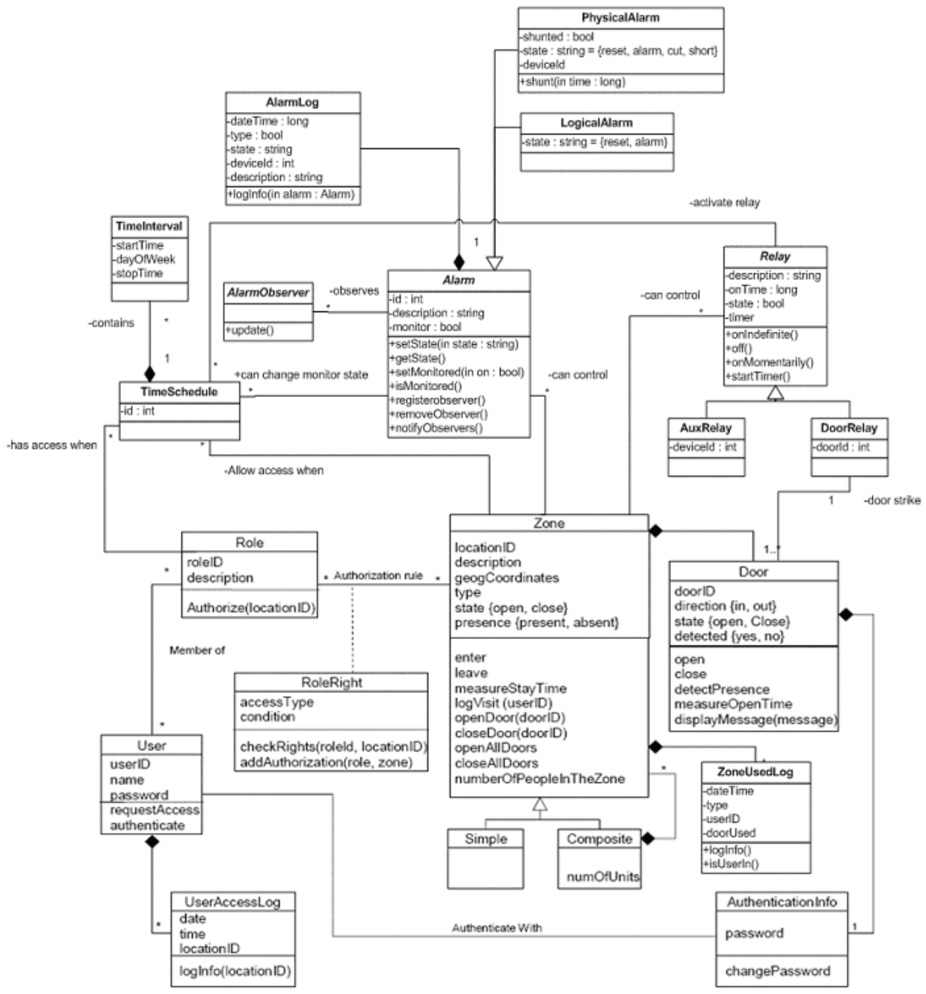


Fig. 5. Class Diagram for Access Control to Physical Structures

Example. Building management wants to put in place an access control system to control access to certain zones and to control who can access the zones. They need to deny all access to certain zones after 5pm. They want to generate alarms when someone tries to access a zone for which they do not have permission and start monitoring alarms for all the exterior doors at 8pm. Moreover, they want to turn on the main door light at 7pm.

Context. Physical environment with access control system where we need to control access and turn on/off devices based on time constrains.

Problem. We need a way to enforce business rules in an access control system that take effect at a given time and day of the week. For example, a user may have different access needs on different days, as in the following example: 08:30 - 17:00 Mon Wed Fri Standard Hours 08:30 - 19:00 Tue Thur Stays Late 2 Days a Week 08:00 - 12:00 half a day on Saturday

The following forces affect the solution:

- We need a way to automatically cancel access for everyone to some areas of a building at given time and day of the week. Some users might have access to some areas only during a time range of the day.
- We need to provide a way to automatically activate devices based on the time and day of the week.
- We need a way to represent a day of the week and time.
- This pattern expands the Access Control to Physical Structures Pattern [9]. Therefore all the forces presented in that pattern are present in this pattern as well.
- We need to restrict access to some users depending on the identity or other characteristics of the visitor.
- The boundaries of the unit of access must be clearly defined.
- There is a variety of users such as employees, contractors, repairpersons, etc., that require access to different parts of a building.
- Since some units will be normally closed, it is necessary to create policies and plans in case of an emergency, such as earthquake or fire.
- We may need to keep track of who entered each unit and when.

Solution. Define the structure of an access control system using an RBAC pattern [7]. Integrate the Alarms Monitoring and Relays patterns and introduce the concept of a time schedule to control when things can/must happen. Time Schedules have two uses; one is to control access times and the other is to configure automatic actions.

*Structure***.** Figure 5 shows a UML class diagram for the Access Control to Physical Structures pattern. We can see how the Alarms Monitoring pattern can be integrated so that **Zones** can control **Alarms**. We also use the Relays pattern so that each door has its own **Door Relay** and the **Aux Relays** can be used to turn on/off devices. Zones can control Relays.

We introduce a **TimeSchedule** class that consists of a few time intervals. A **Time Interval** consists of a start time, a stop time, and selected day of the week. When the system clock activates a Time Schedule, it can automatically perform some actions. Relays can be turned on and off, alarm zones can be activated or deactivated, and doors can be unlocked. To control access times, roles are combined with Time Schedule to determine both where and when a user can gain access to zones and we also need to be able to assign a Time Schedule for the entire zone that applies to all users.

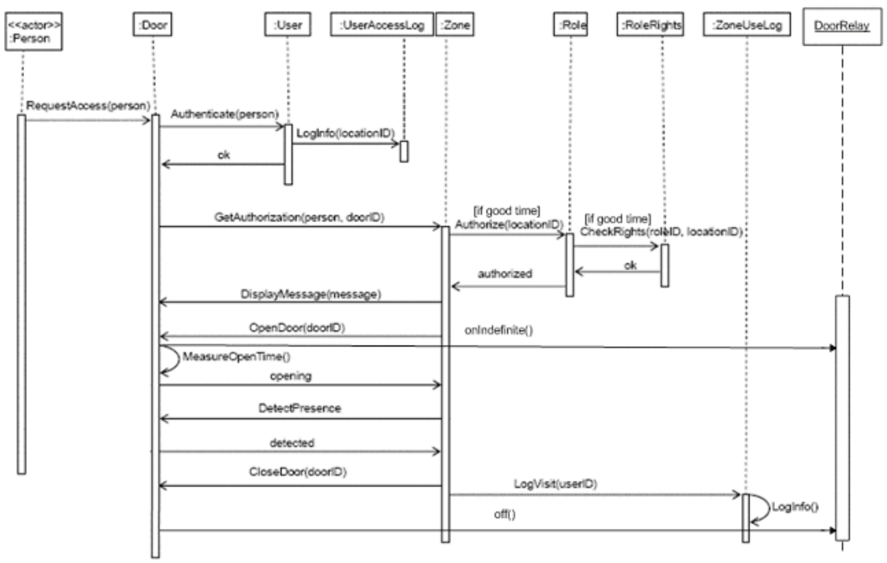


Fig. 6. Sequence Diagram for Entering a Zone

Dynamics. Figure 6 shows a main success scenario for the use case "enter a zone". When the control is passed to the zone, the Zone and the Role will consider their Time Schedules before authorizing a person. Also when the Zone opens or closes a door, the Door calls its Relay to perform the action. Other use cases include: "zone access denied by zone time schedule", "role access denied by time schedule", and "activate alarm by role access denied"; they are not shown for conciseness.

Implementation. Access control systems use centralized processing, distributed processing, or hybrid arrangement. The system architecture should be taken into consideration when designing an access control system, since it can have a significant effect upon operation during a catastrophic system failure.

Centralized Processing. In computer dependent processing systems, all events are gathered by the field panels, and are then sent to the computer for processing. For example if a credential is presented at a door, the door sends the credentials to the central computer or processor. The computer checks the credential against its programming and determines if it is allowed through that door at that time. If valid the computer sends a command back to the panel to release the door. In these systems if the computer goes down, or if communications between the panel and computer is lost, the system can no longer function, to verify proper access, and to process alarms.

Distributed Processing. In distributed processing systems the database is loaded to the field panel. All decisions are made at the field panel and are passed to the computer or logging printer for storage. In these systems if communications is lost, access control continues uninhibited. Furthermore, the events can sometimes be stored in the panels, and can be sent up to the computer once communications is restored. Due to their architecture, systems which employ distributed processing generally offer better reliability, and faster response than systems that rely on central computers for all decision making.

Example Resolved. Building management can configure time schedules and assign them to Zones and Roles, that a way a Zone could have a time schedule from 8-5pm. Moreover, time schedules can be assigned to Relays and Alarms so that they can be activated and monitored respectively when the system clock activates these time schedules.

Known Uses. Many commercial or institutional buildings control access to restricted units using the concepts presented here. This model corresponds to an architecture that is seen in commercial products, such as: Secure Perfect, Picture Perfect and Diamond II. BACnet is a standard that includes access control to buildings and incorporates RBAC, zones, and schedules [12].

Consequences. They are a combination of the previous patterns.

Related Patterns. This pattern is a combination of the RBAC (or another suitable authorization model [7]), the Physical Structures pattern, the Access Control for Physical Structures pattern, the Alarm Monitoring pattern, and the Relays pattern.

4 Related Work and Discussion

There is a considerable amount of work on the security of SCADA (Supervisory Control and Data Acquisition) networks, e.g. [13][2]. However, that type of security is not applicable to physical protection, SCADA security applies only to the information functions of these networks. Building security appears as a neglected area, the only analysis of this problem come from industry white papers, e.g. [14], which emphasizes the need for the convergence of physical and logical security. Some documents about BACnet discuss its rationale, which has some close aspects to our work, they use RBAC and try to derive conceptual models of their protocols but they are tied to some implementation details. R.Martin's book [15] has a detailed building security system used as an example of applying object-oriented methods, but he does not use patterns.

Some work attempts to find ways to control location information of people, e.g. [16]. A model of that type can be made more precise by specifying more closely the location information, as we can do with our patterns. [17] proposes a formal model for the release of information about user location in a smart building while respecting privacy constraints. Our approach can be used to define location of sensors and for interpreting information about user location. By defining more closely the location of a user, her privacy can be protected more precisely. Our finer location definition comes from the fact that we use the Composite pattern [4], a recursive structure that can have any levels of containment. This approach can also be combined with approaches that use geometric models of location [18] by providing an anchor or context for the location information.

There is also much work on context-dependent security, e.g. [19][20], where access to services or resources depends on the location of the user. It is clear that our model can make contexts more specific by defining them in relation to zones or access points (gates). Context models are usually applied in wireless environments, a user could prove wirelessly that he is in front of a gate and that gate could be opened remotely. In the literature, the rights considered are about information, e.g. to a list of nearby restaurants, our approach can unify both types of accesses.

Emergency situations are very important for physical access, in the case of fire all doors should be opened, in the case of an attack as the recent one at Virginia Tech, all doors could be closed. Our model handles these cases very well, all the doors are just instances of a class Door and we can have operations such as 'open all doors' or 'close all doors', that apply to all the objects of that class.

As indicated earlier, the presented model is semi-formal. It can be made more formal by adding Object Constraint Language (OCL) constraints [21]. OCL can be used to add formal annotations to a UML model. Another possibility is the use of the template notation of [22]. In that way, we can at least define more precise requirements and maybe prove properties of some sections of the system.

5 Conclusions

We have described using patterns the basic features and concepts that any modern Physical Access Control system must have. As indicated, these patterns can guide the design of physical access control systems or they can be used to evaluate current products of this type. Other possibilities include the dynamic restriction of the locations where a suspicious user could go or reconfiguration of exits in case of emergencies. They can also be used in conjunction with privacy-oriented models. Another next step is a pattern that could be more event-driven in a way that any subsystem that generates events could be hooked dynamically. Our Access Control system could be one of the subsystems as well as a video system, a metal detector, a drug detector, and so forth.

The contributions of this paper include a constructive semi-formal model of access control to physical structures and a set of patterns, which can be used on their own to build other models. This model also illustrates composition of features by composing patterns. Our future work will include combining this model with identity management patterns [23], with context-based models, and with traditional RBAC models (users could have their access to information and to physical locations defined by the same model). Further formalization is another aspect we are considering. Finally, physical accesses may involve not just entering a zone but may include physical removal of specific objects identified by RFID devices.

References

1. SSPC135/LSS-WG: Physical access control with BACnet (October 2006), http://www.bacnet.org/bibliography/bac-10-06.pdf
2. Majdalawieh, M., Parisi-Presicce, F., Wijesekera, D.: Dnpsec: A security framework for dnp3 in SCADA systems. In: Internat. Joint Conf. on Computer Information and Systems Sciences and Engineering, Bridgeport, CT (December 10-20, 2005)
3. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture: A System of Patterns, vol. 1. Wiley, Chichester (1996)
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston, Mass (1994)
5. Schumacher, M., Fernandez, E.B., Hybertson, D., Buschmann, F., Sommerlad, P.: Security Patterns: Integrating security and systems engineering. J. Wiley & Sons, Chichester (2006)
6. Steel, C., Nagappan, R., Lai, R.: Core Security Patterns: Best Strategies for J2EE, Web Services, and Identity Management. Prentice Hall, Upper Saddle River, New Jersey (2005)
7. Fernandez, E.B., Pan, R.: A pattern language for security models. In: Procs, of PLoP (2001), http://hillside.net/plop/plop2001/accepted_submissions/accepted-papers.html
8. Priebe, T., Fernandez, E.B., Mehlau, J.I., Pernul, G.: A pattern system for access control. In: Procs. of the 18th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Sitges, Spain, pp. 235–249 (July 2004)
9. Desouza-Doucet, A.: Controlling access to physical locations, M.S. Thesis, dept. of computer science and eng., Florida Atlantic University (April 2006)
10. Fernandez, E.B., Sinibaldi, J.C.: More patterns for operating system access control. In: Proc. of the 8th European conference on Pattern Languages of Programs, pp. 381–398 (2003), http://hillside.net/europlop
11. Fernandez, E.B.: Security patterns (keynote talk and paper). In: Procs. of the Eigth International Symposium on System and Information Security - SSI2006, Sao Jose dos Campos, Brazil (November 08-10, 2006)
12. Ritter, D., Isler, B., Mundt, H., Treado, S.: Access control in bacnet. BACnet today (supplement to ASHRAE Journal) B26–B32 (November 2006)
13. Chandia, R., Gonzalez, J., Kilpatrick, T., Papa, M., Shenoi, S.: Security strategies for SCADA networks. In: Procs. First Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection,
14. Bridging the great divide: The convergence of physical and logical security (August 2006), Imprivata(http://www.imprivata.com)
15. Martin, R.: In Designing Object-Oriented C++ Applications Using the Booch Method (Chapter 6). Prentice-Hall, Englewood Cliffs (1995)
16. Hengartner, U., Steenkiste, P.: Implementing access control to people location information. In: Procs. of the ACM Symposium on Access Control Models and Technologies (SACMAT'04), ACM Press, New York (2004)

17. Boyer, J., Tan, K., Gunter, C.: Privacy-sensitive location information systems in smart buildings. In: Procs. of the 3rd Int. Conf. on Security for Pervasive Computing, York, England (April 2006)
18. Atluri, V., Shin, H.: Efficient enforcement of security policies based on tracking of mobile users. In: 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (July 2006)
19. Corradi, A., Montanari, R., Tibaldi, D.: Context-based access control management in ubiquitous environments. In: Proceedings of the Third IEEE International Symposium on Network Computing and Applications (NCA'04), IEEE Computer Society Press, Boston, MA (August 30 - September 01, 2004)
20. Huldenbosch, R., Salden, A., Bargh, M.S., Ebben, P.W.G., Reitsma, J.: Context-sensitive access control. In: Procs. of SACMAT (2005) 111–119 (2005)
21. Warmer, J., Kleppe, A.: The Object Constraint Language, 2nd edn. Addison-Wesley, Reading (2003)
22. Ray, I., Li, N., Kim, D., France, R.: Using parameterized UML to specify and compose access control models. In: Proceedings of the Sixth IFIP WG 11.5 Conference on Integrity and Control in Information Systems. Lausanne, Switzerland (November 2003)
23. Delessy, N., Fernandez, E.B., Larrondo-Petrie, M.: A pattern language for identity management. In: Delessy, N. (ed.) Accepted for the 2nd IARIA Int. Multiconference on Computing in the Global Information Technology (ICCGI 2007), Guadeloupe, French Caribbean (March 4-9, 2007)

XACML Policies for Exclusive Resource Usage

Vijayant Dhankhar, Saket Kaushik, and Duminda Wijesekera

Department of Information & Software Engineering
George Mason University
Fairfax, VA 22030, U.S.A
{vdhankha,skaushik,dwijesek}@gmu.edu

Abstract. The *extensible access control markup language (XACML)* is the standard access control policy specification language of the World Wide Web. XACML does not provide exclusive accesses to globally resources. We do so by enhancing the policy execution framework with locks.

1 Introduction

The *extensible access control markup language (XACML)* [18] is the standard language to specify accesses to resources available on the world wide web. However, the XACML normative specifications lack necessary syntax to specify *exclusive access* to resources, and furthermore, publicly available XACML policy enforcement frameworks do not enforce them. Given that web orchestrations are composed from existing ones using languages such as BPEL [16] may pose concurrent request for exclusively usable resources (such as updating an XML schema), we enhance XACML syntax and enforcement mechanisms to do so.

Perils of not using a synchronization mechanism (such as the *dirty read* [21] in distributed systems) in exclusive accesses are well known. Consequently, we advocate to make a distinction in granting exclusive access and non-exclusive accesses by access controllers. Thus, we add appropriate syntax to XACML and an enforcement mechanism using locks. Consequently, if and when granted, the access control policy is aware that such permissions are granted in exclusion. This enrichment to XACML has no relationship to application level concurrency control, but not surprisingly, due to the enforced semantic distinction between exclusive and non exclusive acccesses, aids in enforcing *separation of duty* principles [12,8,9,20].

To enforce enhanced XACML policies, we add a lock manager to the policy enforcement module and require that all globally accessible resources register with a unique lock manager. In order to ensure starvation avoidance, we assume that resource requesters give up such resource after their usage - although this latter aspect is being driven by policy in our ongoing work.

The rest of the paper is organized as follows. Section 2 has related work. Section 3 presents sample Use Cases and Misuse Cases of exclusive access and our design to realize the former. Syntactic extensions to XACML appear in Section 4 and Section 5 describes the architectural enhancements used to enforce locking. Section 6 describes our implementation and Section 7 informally argues that our locking ensures safety and liveliness. Finally, Section 8 concludes the paper.

2 Related Work

Motivated by a desire to to introduce trust-based, context-aware access control framework for Web Service invocations, including support for RBAC sessions, Bhatti *et al.*, [4,6,5,7] define X-RBAC and X-GTRBAC models for access control frameworks for Web Services, respectively based on RBAC [12] and GTRBAC [14] models of access control. However, they do not provide mechanisms to enforce dynamic separation of duty (DSoD) policies, as is the case with current XACML RBAC profile [17]. Cardea by Lepro *et al.* [15] offers a dynamic access control system for the Web, where the *dynamism* means that the request is not bound to local identities at runtime, but instead uses a remote requester's context instead. However, Cardea does not explicitly address concurrent access to exclusively used resources nor dynamic separation of duty policies.

3 Use Cases, Misuse Cases and Requirements

Although some existing work on Web Services orchestration argues the need to lock shared resources [3,13], to the best of our knowledge they only reserving syntax for locks [3]. Following Use Cases show the need.

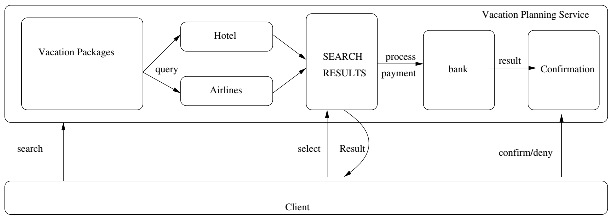


Fig. 1. Vacation Planning Service

3.1 Use Case 1: Exclusive Access

Consider an example, *vacation planning service (VPlanner)* that reserves hotel rooms and air tickets for its clients, whose work-flow is given in figure 1 [23]. As seen, this interactive service is used to first *searches* for available rooms and air tickets for specified dates and destinations and presents various alternatives to its clients, from which the latter *chooses* alternative for reservations. The service then *initiates* a monetary transfer request to the credit granting agency. On success, the room(s) and air tickets are reserved and aborted otherwise. An efficient implementation should invoke airline and hotel room searches concurrently, while, work-flow dependencies require that monetary transfer request should wait till other parts of the procedure are complete.

Now, suppose two clients are searching for reservations from the VPlanner and are shown the same tickets and hotel rooms. This is potentially dangerous because both can choose the same room or ticket, where simultaneous requests can deadlock two BPEL server processes. One way of avoiding this situation is to not offer a second client the choices while a precedent client s in the process of reserving a package - thus requiring the VPlanner to lock rooms and tickets during ongoing reservations, referred to as *tentative locking of resources* in the Web Services literature [3].

3.2 Use Case 2: Enforcing Dynamic Constraints

Example 1 (DSoD [8]). Consider a DSoD constraint *an employee cannot invoke role 1 in a session if another role, role 2, is already invoked in some other session. Assuming a data structure maintained by the system* 'sessions' with following XML schema:

```
<user id="ID">
  <sessions>
    <session id="123123">
       <role name="role1"/>
       <role name="role3"/>
    </session> ...
  </sessions>
</user>
```

An abbreviated DSoD XACML policy as follows:

```
1<Rule RuleId="DSoD:role1-role2:requirements" Effect="Deny">
2 <!-- SoD Rule for Example 1 (begin) -->
3 <Target>
4   <Subjects>
5     <AnySubject/>
6   </Subjects>
7   <Resources>
8     <AnyResource/>
9   </Resources>
10  <Actions>
11    <Action>
12      <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
13        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">activate-role</AttributeValue>
14        <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string" AttributeId="urn:oasis:names:
              tc:xacml:1.0:action:action-id"/>
15      </ActionMatch>
16    </Action>
17  </Actions>
18 </Target>
19 <!-- SOD check -->
20 <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:seperationOfDutyCheck">
21  <!-- sessions -->
22  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:getSubjectSessions">
23    <!-- subject-id -->
24    <AttributeSelector RequestContextPath="//Request/Subject/Attribute[1]/AttributeValue/text()" DataType="http://
          www.w3.org/2001/XMLSchema#string"/>
25  </Apply>
26  <!-- role-id -->
27  <AttributeSelector RequestContextPath="//Request/Resource/Attribute[2]/AttributeValue/text()" DataType="http://
         www.w3.org/2001/XMLSchema#string"/>
28  <!-- comma delimited set of conflicting roles -->
29  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">role1,role2</AttributeValue>
30 </Condition>
31</Rule>
```

Policy 1. DSoD policy

Example 1 above is a DSoD policy expressed in terms of the XACML RBAC profile [17], where as stated in lines 20-30, role 1 and role 2 cannot be co-activated. However, this policy is not currently enforceable because XACML enforcement does not consider concurrent requests. To be fair, the XACML RBAC profile *outsources* the process of enabling roles to the *Role Enablement Authority* module.

Our design enables the follwing Use cases:

Secure registration of resources: A resource may register itself to a unique lock manager.

Secure deregistration of resources: Only a resource is able to securely deregister itself from the lock manager.

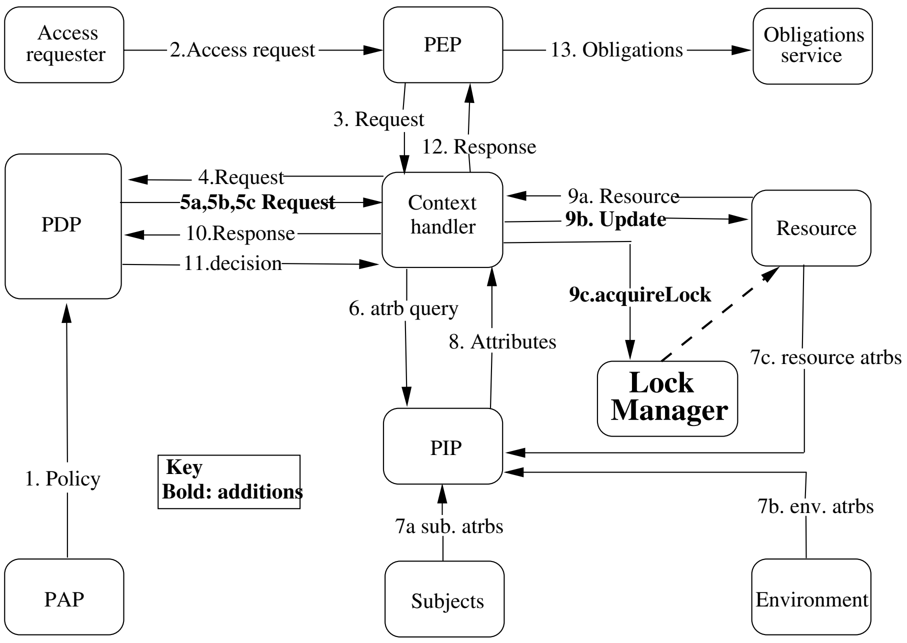


Fig. 2. Extended XACML data flow diagram

Exclusive access /relinquish resources: Exclusive use of a resource must be granted to a unique requester at any given time.

3.3 Preventing Misuse Cases

Our design prevents following Misuse Cases:

Registering a resource with multiple lock managers: An exclusively usable resource being registered with multiple lock managers, referred to as *singular registration*.
Spoofing a resource: Others (de)registering an exclusively accessible resource.
Preventing simultaneous exclusive access: Multiple requesters simultaneous accessing an exclusively usable resource.
Starvation: Refusing exclusive access to resources when not in use.

4 Enhancing the XACML Syntax

Because our solution use locks, we add them to XACML syntax. Each of the following elements are specified within `<Rule/>`, `<Policy/>` and `<PolicySet/>` elements of XACML.

– `<PreAction />` specifies a set of locks to be acquired before rule evaluation.
  `<AcquireLocks />` specifies a set of locks to be acquired and is a sub element of `<PreAction/>` element, where the `<AcquireLock>` sub element specifies an individual lock.

– `<PostAction />` identifies a set of actions to be performed after (a) rule evaluation leads to a permitted request, (b) rule evaluation leads to a denied request. The set of actions may include releasing locks or updating system resources. For different evaluation results multiple `<PostAction />` elements may be defined.

  `Effect` attribute indicate the *effect* of a post action, as discussed above.

  `<Updates />` specifies updates to be performed in a `<PostAction/>`. `<Update>` sub element specifies an individual change.

  `<ReleaseLocks />` specifies a set of locks to release and is a sub element of `<PostAction/>` element. `<ReleaseLock>` sub element specifies an individual lock.

Introduced elements specify lock acquisition prerequisite for evaluating a rule and post evaluation steps to be taken. The following example policy extends policy 1 with proposed syntactic enhancements.

```
1<Rule RuleId="DSoD:role1-role2:requirements" Effect="Deny">
2  <PreAction>
3    <AquireLocks>
4      <AquireLock>
5        <!-- sessions -->
6        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:getSubjectSessions">
7          <!-- subject-id -->
8          <AttributeSelector RequestContextPath="//Request/Subject/Attribute[1]/AttributeValue/text()" DataType="http
                 ://www.w3.org/2001/XMLSchema#string"/>
9        </Apply>
10     </AquireLock>
11     .
12     .
13     .
14   </AquireLocks>
15 </PreAction>
16     .
17     . <!-- DSoD Rule in Example 1 -->
18     .
19 <PostAction Effect="Permit">
20   <Updates>
21     <Update>
22       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:addRoleToSession">
23          <!-- role-id -->
24         <AttributeSelector RequestContextPath="//Request/Resource/Attribute[2]/AttributeValue/text()" DataType="
                 http://www.w3.org/2001/XMLSchema#string"/>
25         <!-- session-id -->
26         <AttributeSelector RequestContextPath="//Request/Resource/Attribute[3]/AttributeValue/text()" DataType="
                 http://www.w3.org/2001/XMLSchema#string"/>
27         <!-- sessions -->
28         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:getSubjectSessions">
29           <!-- subject-id -->
30           <AttributeSelector RequestContextPath="//Request/Subject/Attribute[1]/AttributeValue/text()" DataType="
                   http://www.w3.org/2001/XMLSchema#string"/>
31         </Apply>
32       </Apply>
33     </Update>
34     .
35     .
36     .
37   </Updates>
38   <ReleaseLocks>
39     <ReleaseLock>
40       <!-- sessions -->
41       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:getSubjectSessions">
42         <!-- subject-id -->
43         <AttributeSelector RequestContextPath="//Request/Subject/Attribute[1]/AttributeValue/text()" DataType="http
                 ://www.w3.org/2001/XMLSchema#string"/>
44       </Apply>
45     </ReleaseLock>
46     .
47     .
48     .
49   </ReleaseLocks>
50 </PostAction>
51</Rule>
```

Policy 3. *Enhancements to DSoD policy*

The `PreAction` element in lines (2–15) of policy 3 above states that before evaluating a rule in the DSoD policy, the user session must be locked (lines 5-10). Similarly, PostAction element in lines (19-50) requires that after the policy has been evaluated, the locks acquired earlier be released for future concurrency-free changes to user sessions. We assume here that resources are not created during XACML evaluation (they already exist and are registered with a lock manager), however their usage status, *i.e.*, open for read/write, *etc.*, may be modified during a policy evaluation. For example, an XACML evaluation can modify a log file, *etc.*

4.1 Implemented Semantics of Syntactic Extensions

Postaction elements are evaluated in the following manner.

- Post action only updates resources for locks obtained at the corresponding level or those obtained at the level of the container.
- If two rules within a policy require same lock then they must be acquired and released at Policy level. Such locks are visible within all embedded rules. Similarly, if a lock is acquired at the `PolicySet` level then it is visible to all embedded policies.
- If a resource must be updated in multiple rules, then corresponding lock must be acquired at their container level, *i.e.*, `Policy`.
- Rule evaluation within a `Policy` element is evaluated by a single thread of execution.
- If locks are required at only the rule level, they must be released at the rule level `<PostAction/>`, otherwise, they must be released at the `<Policy/>` level `<PostAction/>`

4.2 XACML Functions

Extensions to policy syntax is achieved with the help of following functions:

function:getSubjectSessions($subject-id as string). Accepts a subject-id as an input and returns a bag of session objects used by this subject. If the subject's sessions have been acquired by PDP through exclusive access, *i.e.*, locked, then the sessions are cached till the lock to the sessions is released.

function:addRoleToSession($role-id as string, $session-id as string, $sessions as bag). Accepts role-id, session-id and a bag of sessions as input and it adds the passed role to the particular session. Sessions data structure contains all the sessions and session-id is used to locate the relevant session in the current implementation, although more efficient implementations are possible.

5 Architecturereal Enhancements Needed for Locks

Figure 2 shows the existing XACML execution model with data flow for policy control [18]. The data flow begins with the Policy Administration Point (PAP) that authors the policies evaluated by the XACML framework, shown in *Flow 1*. Next, access requests (*Flow 2*), initiated by resource requesters, are intercepted by the Policy Enforcement Point (PEP). PEP forwards them, *Flow 3*, to the Context Handler (CH) with optional requester attributes and environmental conditions required for processing. Context handler has following three functions:

Flow 4: Translate access requests into a format understood by the Policy Decision Point (PDP).

Flow 10: In response to Flow 5, generate the evaluation context by gathering resource, requester attributes and current system state of from the policy information point (PIP) (i.e. from, Flows 6,7,8 and 9), and pass them to the PDP.

Flow 12: Receive policy decisions from the PDP and translate them back to the PEP.

PDP evaluates an XACML policy applicable to the access request and accompanying context. If fails, the access is denied and granted otherwise. This decision is made available to the context handler of Flow 11 and is relayed to the PEP for enforcement.

Figure 2 shows the *extended* XACML data flow diagram that introduces a *lock manager (LM)* to augment XACML access control decisions. The *Lock Manager* grants and revokes locks for accessing resources registered with itself, requiring extra data flows as follows:

Flow 5b - Update System Request (USR): May be initiated by the PDP to update system resources for setting up an access. For example, enabling a role may require that the user session (a system resource) be updated.

Flow 5c - Create Lock Request (CRL): Is initiated by the PDP on behalf of the requesting process, in response for exclusive access to an available resource. This is finally refined to the `acquireLock` operation (9c.), where the lock is *owned* by the requesting process.

Flow 10 - Response to PDP queries (overloaded): We reuse the response sent by the context handler to the PDP queries for sending USR and CRL responses in addition to resource query response.

Flow 9b - Resource update, 9b: Upon enforcing the access control decision, the PEP updates an internal log of accesses. This data flow enables enforcing history based resource usage.

Flow 9c - AcquireLock, 9c: Instructs the LM to invoke a lock on behalf of a requester. Based on the availability of a resource, this operation may succeed or fail.

As in the normative XACML specification, we assume that all attributes have been authenticated (using attribute certificate authenticity) prior to policy evaluation. We discuss the additional complexity due to these addition later in section 5.2. First, we describe the Lock Manager design.

5.1 Lock Manager (LM)

The Lock Manager (see figure 2) is a *privileged process* that, at any given time, has only a single instance running. The Lock Manager maintains and creates locks for resources it manages. The functionality of the Lock Manager that provides and maintains locks are as follows:

Lock Acquisition. Lock acquisition is an atomic operation `acquireLock`, implemented using an atomic operation, akin to the unix `test&set` operation [22], with two associated strings - `requesterId` and `resourceId` with the `acquireLock`. These strings can be qualified names or URIs of network entities. The actual call is made by the lock manager within a critical section, as shown in section 6.

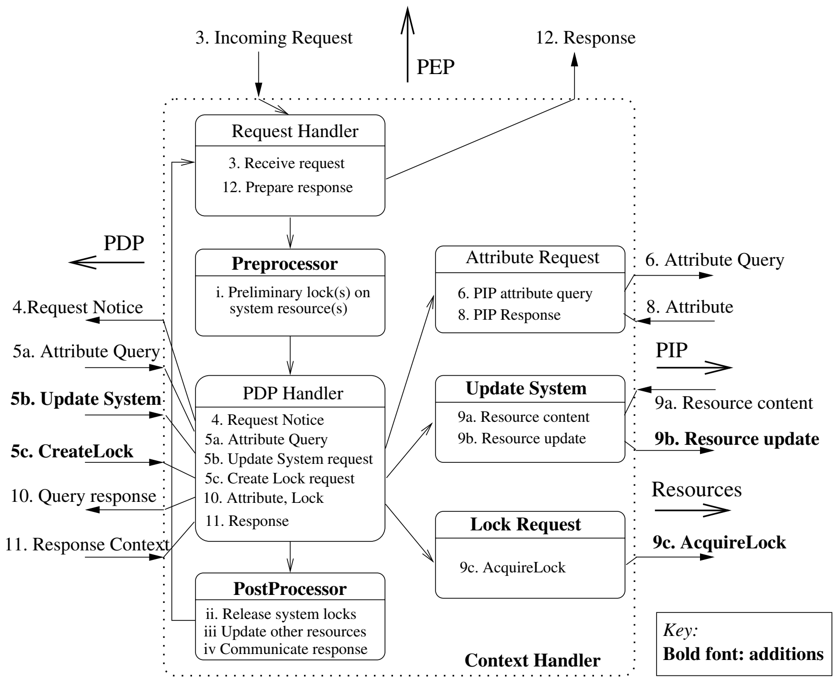


Fig. 3. Context Handler

Releasing a Lock. Lock release is an atomic process implemented as the method *releaseLock*, implemented atomically.

Verifying a Lock. The `verifyLock` operation verifies the validity of a lock. The execution model states that for every resource update must be preceded by a call to *verifyLock*, and is invoked by the resource manager for verifying the validity of a lock.

The execution model is that a requester gets the lock and at the time of resource usage and presents the locking permission to the resource manager that in turn verifies the validity of the presenter's claim with the lock manager and or PDP.

Registering with a Single LM. Registering a resource with a single lock manager is a basic design requirement enforced by requiring an attribute certificate where each resource is bound to a *single* lock manager (by a local certificate authority) using an X.509 attribute certificate [11].

5.2 Enhancing the Context Handler

We enhance the context handler with resource pool bootstrapping, termination and maintenance, (*i.e.*, performing singular registration, deregistration, *etc.*) and extending the business logic for additional functionality.

Resource Pool Maintenance. We extend the context handler to support the lock manager bootstrapping and maintenance of the resource pool. Algorithms in this section are written in pseudo code, with '–>' symbol indicating a call to a sub-module within the context handler.

Securely Registering Resources. To prevent multiple concurrent registration requests being invoked by a resource, we introduce the secure registration of resource procedure. Secure registration begins with a registration request by a resource, accompanied with a verifiable attribute certificate containing the resource and the lock manager. This request is serviced by Algorithm 1 below, as shown in figure 3.

```
 1 register(R,C,LM)
 2 Inputs: R(resource), C(certificate), LM(lock manager)
 3 Output: Lock information, Exception
 4
 5 Translate request to XML document
 6 Pass translated XML to PDP Handler
 7 -->Invoke PDP to verify R,C and LM
 8  if (decision == accept)
 9    Lock Request for lock to global.lock
10    -->Acquire lock to global.lock
11    Update System Request for variable
12    -->Create variable lock.R
13      if (lock.R NOT IN global.lock)//
14        Assign lock.R.subjectID = ""
15        Insert lock.R in global.lock
16        Assign message = lock.R
17        Hand request to PostProcessor
18        -->Lock Request for release lock
19           -->Release lock
20           Return request+message to Request Handler
21      else
22        Assign exception = Already Registered
23        Hand Request to PostProcessor
24        -->Lock Request for release lock
25           -->Release lock
26           Return request+exception to Request Handler
27  else
28    Assign exception = Invalid Request
29    Hand the request to PostProcessor
30    -->Return request+exception to Request Handler
```

Algorithm 1. *Secure Registration*

The `register` method accepts three inputs – R, the registering resource; C, its attribute certificate, and LM, the Lock Manager. This request is handed to the 'Request handler' module of CH by the PEP (line 5). The request handler translates the request into XML and hands it to PDP handler for further processing (line 6). The PDP handler invokes the PDP and processes the response (lines 7,8). If the decision is accept, a lock is acquired (lines 9,10) and an update request is fired (line 11). Based on the success of this call, lock is release and the relevant message is returned to the request handler (lines 18-26). If PDP denies the register request, an appropriate response is constructed as well (line 27-30).

Securely Deregistering a Resource. Similar to registration, deregistration requires locking support because in order to prevent being deregistered while in use. The process flow is as follows:

```
 1 deregister(R,C,LM)
 2 Inputs: R(resource), C(certificate), LM(lock manager)
 3 Output: boolean, Exception
 4
 5 Translate request to XML document
 6 Pass translated XML to PDP Handler
 7 -->Invoke PDP to verify R,C and LM
 8  if (decision == accept)
 9   Lock Request for lock to global.lock
10   -->Acquire lock to global.lock
11   if (acquirelock()== false)
12    Assign exception = In use // no waiting!!
13    Hand request to PostProcessor
14    -->Send request+exception to Request Handler
15   else
16    if(lock.R IN global.lock && lock.R.subjectId="")
17     Update System Request
18     -->Assign global.lock = global.lock -lock.R
19       Assign message = true
20     Hand request to PostProcessor
21     -->Lock Request for release lock
22       -->Release Lock // will succeed
23       Return request+message to Request Handler
24    else
25     Assign exception = Does not exist/In use
26     Hand Request to PostProcessor
27    -->Lock Request for release lock
28      -->Release lock
29      Return request+exception to Request Handler
30  else
31   Assign exception = Invalid Request
32   Hand the request to PostProcessor
33   -->Return request+exception to Request Handler
```

Algorithm 2. *Secure Deregistration*

Similar to the `register` method `deregister` accepts the same three inputs and securely removes the resource from LM. The difference here is that the update request removes the lock from a global lock data structure in a critical section (line 18).

Extending the Business Logic

Gaining Exclusive Access to Resources. Once a resource is registered, exclusive access to it can be guaranteed by a process very similar to the above processes, as follows:

```
exclusiveAccess(R,S)
Input: R (resource), S (Subject)
Output: boolean, Exception

Translate request to XML document
Pass translated XML to Pre processor
-->Lock Request for lock to global.lock
 -->Acquire lock to global.lock
 if (acquireLock() == false)
  Assign exception = In use // no wait
  Hand Request to PostProcessor
  -->Send Request+exception to Request Handler
 else
  Hand request to PDP Handler
  -->Invoke PDP for authorizing S to R
  -->Attribute Query for lock.R
    -->Read lock.R
     if(lock.R IN global.lock)
      send lock.R to PDP
     else
      exception = resource unknown
      Hand request to PostProcessor
      -->Return Request+exception // to RH
   if (decision == accept)
    Update System Request
    -->Assign lock.R.subjectId = S
      Assign message = true //
    Hand request to PostProcessor
    -->Lock Request for release lock
     -->Release Lock
     Return request+message to Request Handler
   else
    Assign exception = Invalid Request
    Hand the request to PostProcessor
    -->Return request+exception to Request Handler
```

Algorithm 3. *Exclusive access*

The `exclusiveAccess` method invokes a similar CH work-flow for granting access to externally usable resources. The main difference here is a call to the pre processor module that acquires locks before beginning any PDP evaluation (lines 7-12). In addition, the PDP may query for lock attributes (lines 16-23). Finally, if the request is granted, the lock manager updates `lock.R` to indicate the new owner (line 26).

Using Resource. Exercising exclusive access, once a requester has gained such an access to a resource, is a call to the resource (through the PEP). It is the resource's responsibility to ensure that the requester owns a valid lock to it, done by a *verifyLock* call to the lock manager (details are omitted due to lack of space).

Guaranteeing Dynamic/ History-Based Access Constraints. Steps for enforcing dynamic constraints are as follows:

```
 1 accessResource(R,S)
 2 Inputs: R (resource), S (Subject)
 3 Output: boolean, Exception
 4
 5 Translate request to XML document
 6 Pass translated XML to Pre processor
 7 -->Locate internal resources for the request
 8   Lock Request for locks to all internal resources
 9   -->while(more resources)
10        { Acquire lock } // locking phase
11   if (all locks acquired == false)
12     Assign exception = Cannot process // no wait
13     Hand Request to PostProcessor
14     -->Send Request+exception to Request Handler
15   else
16     Invoke PDP for authorizing S to R
17     Attribute Queries (optionally)
18     -->Read attribute
19        if(attribute present)
20          send attribute to PDP
21        else
22          exception = resource unknown
23          Hand request to PostProcessor
24          -->Send Request+exception to Request Handler
25     if (decision == accept)
26       Update System Request
27       -->while(more resources need updation)
28            { update ith resource }
29          Assign message = true //
30       Hand request to PostProcessor
31       -->Lock Request for releasing all locks
32         -->Release Lock
33         Return request+message to Request Handler
34     else
35       Assign exception = Invalid Request
36       Hand the request to PostProcessor
37       -->Return request+exception to Request Handler
```

Algorithm 4. *Dynamic constraints*

Releasing Resource. A resource requester holding a lock can *release* the resource. A simple modification to the algorithms presented above does this, where the details are omitted due to lack of space.

6 Implementing the Enhanced Design

In this section presents the salient features of our LM implementation. We begin our discussion with data structures to implement locks, followed by a sample Java snippet and WSDL interaction to acquire locks.

```
1 <xs:element name="Lock"
2  type="xacml-context:LockType"/>
3 <xs:complexType name="LockType">
4   <xs:sequence>
5     <xs:element name="resourceId" type="xs:string"
6       minOccurs="1" maxOccurs="1"/>
7     <xs:element name="ownerId" type="xs:string"
8       minOccurs="0" maxOccurs="1"/>
9   </xs:sequence>
10 </xs:complexType>
```

Listing 1. *The Lock Data Structure*

Listing 1 shows the *LockType* XML Schema (lines 3-10) that represents a lock for a *single* shared resource. The data structure identifies the lock through a `resourceID` and an `ownerId`. An available lock has an `ownerId` as an empty string, while a locked resource has a non-empty `ownerId`. Several such locks (one for each shared resource) are stored in a global LM-data structure called the *GlobalLock*.

```
<xs:element name="GlobalLock"
 type="xacml-context:GlobalLockType"/>
<xs:complexType name="GlobalLockType">
 <xs:sequence>
   <xs:element ref="xacml-context:GlobalLockEntryType"
     maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
```

Listing 2. *The Global Lock Data Structure*

```
<xs:complexType name="GlobalLockEntryType">
 <xs:sequence>
   <xs:element name="resource" type="xacml-context:Resource"
     minOccurs="1" maxOccurs="1"/>
   <xs:element name="lock" type="xacml-context:Lock"
     minOccurs="0" maxOccurs="1"/>
 </xs:sequence>
</xs:complexType>
```

Listing 3. *The Global Lock Entry Type*

The *Global Lock Entry Type* data structure stores (key,value) pairs including resource and its locks.

```
<wsdl:message name="aquireLockRequest">
  <wsdl:part name="correlationSet" element=
      "xacml-context:CorrelationSet" />
  <wsdl:part name="lock" element=
      "xacml-context:Lock" />
</wsdl:message>

<wsdl:message name="lockResult">
  <wsdl:part name="correlationSet" element=
      "xacml-context:CorrelationSet" />
  <wsdl:part name="result" element=
      "xacml-context:LockResult" />
</wsdl:message>
```

Listing 4. *Lock acquisition/response message*

Listing 3 shows WSDL message definitions for the `acquireLock` request and the *result* response.

```
1 public class LockManager {
2
3 // list of all the resources and locks
4 public static Map GlobalLock = new HashMap();
5
6 /**
7 * Aquire a lock
8 */
9 public static LockResult aquireLock (AquireLockRequest request) {
10
11 Lock lock = null;
12 boolean aquireSuccess = false;
13
14 // aquire global lock
15 synchronized (GlobalLock) {
16
17   // if lock already acquired or not registered then fail
18   if (GlobalLock.containsKey(request.getResourceId())) {
19     lock = GlobalLock.get(request.getResourceId());
20     if (lock==null) {
21       // if no locks exist on the resource
22       // then create a new resource
23       lock = new Lock();
24       lock.setOwnerId(request.getActorId());
25       lock.setResourceId(request.getResourceId());
26       GlobalLock.put(request.getResourceId(), lock);
27       aquireSuccess = true;
28     }
29   }
30
31 } // release the global lock
32
33 LockResult result = new LockResult();
34 if (!aquireSuccess) {
35   // couldnt aquire lock
36   result.setStatus("fail");
37 } else {
38   // lock aquired
39   result.setStatus("pass");
40   result.setLock(lock);
41 }
42 return result;
43}
```

Listing 5. *AcquireLock method*

Listing 5 shows a `Java` implementation of the `AcquireLock` method for acquiring a lock to an existing resource. Java allows *synchronization* through exclusive access to objects that we leverage upon in this implementation (line 17). In Line 16 we aquire exclusive access to the GlobalLock data structure till Line 33. In line 19 we check if the resource is registered with the Lock Manager. Line 21: We check if the lock has already been granted, if not then we create a new Lock with the owner and resource specified in request and add it to GlobalLock (Line 24-27). Finally after updating the GlobalLock we remove our exclusive access to it (Line 33) Line 35: we create a result data structure. Line 36-43: we construct the result for the request accordingly and return it in line 44.

7 Safety and Liveliness Properties

Because we allow concurrent requests and use locks to serialize access to critical sections of the security monitor, we ensure liveliness and safety properties. In this section, we informally argue for them.

Lemma 1. *Given a resource pool $\mathcal{R}$ and a set of lock managers $\mathcal{L}$, a resource $R_i \in \mathcal{R}$ can only be registered with a single lock manager $L_j \in \mathcal{L}$*

Proof Sketch: See [10] for proof.

Lemma 2. *Given a resource R_i and a lock manager L_K to which R_i can register itself,* `register` *method ensures that R_i can be registered only once with L_K.*

Proof Sketch: See [10] for proof.

Lemma 3. *Given a resource R_i and a lock manager L_K to which R_i is registered,* `deregister` *method ensures that only R_i can be deregister itself from L_K.*

Proof Sketch: See [10] for proof.

Theorem 1 (Safety of the exclusive access:). *Given an XACML policy P for exclusive access to an available resource R and multiple concurrent access requests from subjects $S_i, i \in [1, n]$ (i.e.,* `exclusiveAccess(`R, S_i`)` *), only one request from the above set is authorized by P.*

Proof: See [10] for proof.

Theorem 2 (Safety of dynamic constraints:). *Given an XACML policy P for dynamic constraint for access to a resource and multiple conflicting access requests* (`accessResource(`R, S_i`)` *), then P authorizes only one request.*

Proof: Similar to that of Theorem 1.

Theorem 3 (Liveliness1:). *Given a resource R and an exclusive use access request by subject S* `exclusiveAccess(R,S)` *), then policy evaluation will release locks to all internal resources irrespective of the access control decision.*

Proof Sketch: See [10] for proof.

Theorem 4 (Liveliness2:). *Given resources x, y and an exclusive use access requests by subject S* `exclusiveAccess(x,S)` *followed by* `exclusiveAccess(y,S)`*) and concurrent exclusive use access requests by subject T* `exclusiveAccess(y,T)` *followed by* `exclusiveAccess(x,T)`*), then at-least one of the exclusive access requests is denied by policy evaluation and all locks acquired for that policy evaluation are released.*

Proof Sketch: See [10] for proof.

8 Conclusions

XACML is the default access control specification language for the World Wide Web [1,2,19]. But XACML does not currently support three types of access control Use Cases, *viz.*, ensuring exclusive access to globally available resources, preventing access to a resource given a concurrent conflicting use of another resource (DSoD constraints), and preventing access to a resource given a history of conflicting access (such as in Chinese Wall constraints). We extend XACML syntax for supporting the above-mentioned use cases, by enhancing the XACML policy enforcement framework with a *lock manager*, to realize the additional Use Cases, and informally argue that safety and liveliness properties are ensured by our implementation.

References

1. Entrust: http://www.entrust.com/
2. Vordel: http://www.vordel.com/
3. Benatallah, B., Casasti, F., Toumani, F., Hamadi, R.: Conceptual modeling of web service conversations. Technical Report HPL-2003-60, HP Laboratories Palo Alto (March 2003)
4. Bhatti, R., Bertino, E., Ghafoor, A.: A trust-based context-aware access control model for web services. In: 2nd IEEE International Conference on Web Services (ICWS), July 2004, IEEE Computer Society Press, Los Alamitos (2004)
5. Bhatti, R., Joshi, J.B.D., Bertino, E., Ghafoor, A.: Access Control in Dynamic XML-Based Web Services using X-RBAC. In: First International Conference on Web Services (ICWS) (June 2003)
6. Bhatti, R., Joshi, J.B.D., Bertino, E., Ghafoor, A.: X-GTRBAC Admin: A Decentralized Administration Model for Enterprise-Wide Access Control. In: 9th ACM Symposium on Access Control Models and Technologies (SACMAT), June 2005, ACM Press, New York (2005)
7. Bhatti, R., Joshi, J.B.D., Bertino, E., Ghafoor, A.: X-GTRBAC:An XML-Based Policy Specification Framework and Architecture for Enterprise-Wide Access Control. ACM Transactions on Information and System Security (TISSEC) 8(2) (2005)
8. Clark, D., Wilson, D.: A comparison of commercial and military computer security policies. In: IEEE Symposium on Security and Privacy, Oakland, April 1987, pp. 184–194. IEEE Computer Society Press, Los Alamitos (1987)
9. Clark, D., Wilson, D.: Evolution of a model for computer integrity. In: Eleventh National Computer Security Conference, Baltimore (October 1988)
10. Dhankhar, V., Kaushik, S., Wijesekera, D.: XACML policies for exclusive resource usage. Technical Report ISE-TR-07-03, ISE Department, George Mason University, Fairfax (April 2007)
11. Farrell, S., Housley, R.: RFC 3281- an internet attribute certificate (April 2002)
12. Ferraiolo, D.F., Sandhu, R., Gavrila, S., Kuhn, D.R., Chandramouli, R.: Proposed nist standard for role-based access control. ACM Transactions on Information and System Security 4(3), 224–274 (2001)
13. Haddad, S., Moreaux, P., Rampacek, S.: Client synthesis for Web Services by way of a timed semantics (ICEIS 06). In: 8th International Conference on Enterprise Information Systems (May 2006)
14. Joshi, J.B., Bertino, E., Latif, U., Ghafoor, A.: A generalized temporal role-based access control model. IEEE Transaction on Knowledge and Data Engineering 17(1) (Janurary 2005)
15. Lepro, R.: Cardea: Dynamic access control in distributed systems. Technical Report NAS-03-020, NASA Advanced Supercomputing (NAS) Division, NASA Ames Research Center, Moffet Field, CA (November 2003)
16. OASIS: Business process execution language for web services (May 2003)
17. OASIS: Core and hierarchical role based access control (rbac) profile of xacml v2.0 (Feburary 2005), http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-specos.pdf
18. OASIS: Extensible access control markup language (Feburary 2005)
19. RFC 2753: A framework for policy-based admission control
20. Sandhu, R.S.: A lattice interpretation of the chinese wall policy. In: Proc. 15th NIST-NCSC National Computer Security Conference, pp. 329–339 (1992)

21. Tanenbaum, A.S., Steen, M.v.: Distributed Systems: Principles and Paradigms. Prentice-Hall, Englewood Cliffs (2002)
22. Tannenbaum, A.S.: Modern operating systems. Prentice-Hall Inc., Englewood Cliffs, NJ (1992)
23. Tartanoglu, F., Issarny, V., Levy, N., Romanovsky, A.: Dependability in the web service architecture. In: ICSE Workshop on Architecting Dependable Systems, Orlando, FL (May 2002)

Author Index