Lecture Notes in Computer Science

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

Editorial Board

David Hutchison
   Lancaster University, UK
Takeo Kanade
   Carnegie Mellon University, Pittsburgh, PA, USA
Josef Kittler
   University of Surrey, Guildford, UK
Jon M. Kleinberg
   Cornell University, Ithaca, NY, USA
Friedemann Mattern
   ETH Zurich, Switzerland
John C. Mitchell
   Stanford University, CA, USA
Moni Naor
   Weizmann Institute of Science, Rehovot, Israel
Oscar Nierstrasz
   University of Bern, Switzerland
C. Pandu Rangan
   Indian Institute of Technology, Madras, India
Bernhard Steffen
   University of Dortmund, Germany
Madhu Sudan
   Massachusetts Institute of Technology, MA, USA
Demetris Terzopoulos
   University of California, Los Angeles, CA, USA
Doug Tygar
   University of California, Berkeley, CA, USA
Moshe Y. Vardi
   Rice University, Houston, TX, USA
Gerhard Weikum
   Max-Planck Institute of Computer Science, Saarbruecken, Germany
Secure Data Management

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

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

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

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

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

July 2007

Willem Jonker
Milan Petković
Organization

Workshop Organizers

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

Program Committee

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

Additional Referees

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

**Access Control**

A Unified Conflict Resolution Algorithm .......................... 1  
*Amir H. Chinaei, Hamid R. Chinaei, and Frank Wm. Tompa*

Multi-layer Audit of Access Rights ................................ 18  
*Birgit Pfitzmann*

Refinement for Administrative Policies .............................. 33  
*M.A.C. Dekker and S. Etalle*

**Database Security**

Authenticating kNN Query Results in Data Publishing .......... 47  
*Weiwei Cheng and Kian-Lee Tan*

Query Rewriting Algorithm Evaluation for XML Security Views .... 64  
*Nataliya Rassadko*

Answering Queries Based on Imprecision and Uncertainty Trade-Offs in Numeric Databases ........................................... 81  
*Alexander Brodsky, Lei Zhang, and Sushil Jajodia*

Architecture for Data Collection in Database Intrusion Detection Systems ................................................................. 96  
*Xin Jin and Sylvia L. Osborn*

Common Secure Index for Conjunctive Keyword-Based Retrieval over Encrypted Data .......................................................... 108  
*Peishun Wang, Huaxiong Wang, and Josef Pieprzyk*

**Privacy Protection**

Generating Microdata with $P$-Sensitive $K$-Anonymity Property .... 124  
*Traian Marius Truta, Alina Campan, and Paul Meyer*

Preventing Privacy-Invasive Software Using Collaborative Reputation Systems ................................................................. 142  
*Martin Boldt, Bengt Carlsson, Tobias Larsson, and Niklas Lindén*

Towards Improved Privacy Policy Coverage in Healthcare Using Policy Refinement ................................................................. 158  
*Rafae Bhatti and Tyrone Grandison*
### Position Papers

Requirements of Secure Storage Systems for Healthcare Records  ........................................ 174  
*Ragib Hasan, Marianne Winslett, and Radu Sion*

An Intrusion Detection System for Detecting Phishing Attacks  ........................................ 181  
*Hasika Pamunuwa, Duminda Wijesekera, and Csilla Farkas*

A Three-Dimensional Conceptual Framework for Database Privacy  .................................. 193  
*Josep Domingo-Ferrer*

Novel RFID Authentication Schemes for Security Enhancement and System Efficiency  ........... 203  
*N.W. Lo and Kuo-Hui Yeh*

### Author Index

................................................................. 213
A Unified Conflict Resolution Algorithm

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

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

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

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

1 Introduction

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

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

![Graph showing subject hierarchy](image)

**Fig. 1.** An example of the subject hierarchy

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

Adopting one simple policy, such as “the most specific authorization takes precedence,” is not sufficient in practice. For instance, such a policy is insufficient where the subject hierarchy is more complex than tree-based structures and therefore, a subject may have more than one “most specific” authorization. For example, in Figure 1, neither $S_2$ nor $S_5$ is more specific to *User*, with respect to the other, since both of them are at the minimum distance of 1 from *User*. Furthermore, there are situations in which the highest authority (not the most specific one) should be the final arbiter. For instance, assume a student is authorized by the university athletic office to referee hockey games on campus (which requires more than 20 hours per week for several weeks); however, he is required by the department not to accept heavy non-departmental tasks (in order to comply with his full-time registration status). In such a case, the university administration may resolve the conflict by deciding to let him to referee for a limited time. To visualize such a case, assume there is an edge from $S_1$ to $S_2$ in Figure 1 and $S_1$ is labeled positively. Representing the student by *User*, the referees group by $S_2$, the members of the department by $S_5$, and the university members by $S_1$, it is apparent that for this enterprise the most global authorization should take precedence in resolving the conflict.
Some have proposed the “negative takes precedence” policy, but this too is not universally acceptable. For instance, conflicts often are resolved by the “majority takes precedence” rule in voting systems. Additionally, the open policy recommends a default positive authorization for subjects which are not explicitly permitted to access a particular object [6, 10]. Therefore, there are applications in which “positive takes precedence.”

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

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

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

1.2 Outline

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

2 Conflict Resolution Models

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

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

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

2.1 Conflict Resolution Policies

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

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

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

Locality Policy. The common mode of this distance-based policy states that the most specific authorization takes precedence. It applies to distributed organizations whose local branches may recognize an exception to a general rule. For instance, a department in a university may admit an outstanding applicant although the general admission requirement is not completely met. Thus, for a given subject, when both positive and negative authorizations can be derived from different ancestors, the one that is closer to the subject wins. Note that the distance between two nodes (subjects) in a directed acyclic graph is measured by computing the shortest directed path.
The locality policy is not deterministic since no authorization wins when the distances are equal. As an alternative for the locality policy, some enterprises might choose “globality”, where the most general authorization takes precedence. One application of this policy is in distributed organizations whose headquarters makes the final decision on a pre-approved task by a local office. For instance, a supreme court may override the appealed decision. For a given subject, when both positive and negative authorizations can be derived from different ancestors, the one that is farther from the subject wins. Similar to the usual locality policy, globality is not deterministic since no authorization may win.

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

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

### 2.2 Combined Strategies

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

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

We recognize that there are also applications in which the default policy is not appropriate. For instance, in determining whether $S_3$ is authorized access according to Figure 1, we may wish to give priority to the explicit authorization on $S_2$ regardless of whether the value assigned to $S_1$ is positive or negative. This can be accomplished in general by omitting the default policy, using locality or majority as desired to arbitrate a value for $S_3$ and using the preferred authorization to assign a value to $S_1$ only afterwards.
Therefore, to achieve a comprehensive framework, we augment the combined strategies illustrated in Figure 2 with making the default policy optional as well. This results in five more combined strategies namely LP, LMP, P, MLP, and MP. Hence, the framework supports 16 more instances; strategies LP, LMP, and MLP generate four instances each, and strategies P and MP generate two instances each.

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

3 Implementation

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

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

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

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

Figure 3 illustrates the result of Steps 1 and 2 for subject \( User \), object \( obj \), and right \textit{read}, illustrated in Figure 1 as the motivating example.

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

Table 1 illustrates the result of authorization propagation for subject \( User \), object \( obj \), and right \textit{read} as represented by Figure 3.
Step 4: Apply a particular conflict resolution strategy to resolve any conflicts and derive a final effective authorization for the triple \(<S_i, o_j, r_k>\).

Table 2 illustrates the result of applying each of the 48 strategy instances (explained in Section 2) to Table 1. For example, D*LMP* is the strategy instance in which first the default policy is applied and every root subject which is null is initialized to +; then, if there is a conflict, the Locality policy ("the most specific authorization takes precedence") is applied; then if there is still a conflict, the Majority policy is applied; and finally, if the conflict is not resolved, the preference policy in which the positive (+) authorization takes precedence is applied. Let’s see what the result of this strategy instance is on Table 1: by applying the default policy D*, all mode d’s are replaced by +; by applying the locality policy, the conflict is not resolved since there are conflicting modes + and - from the shortest distance 1; however, by applying the majority rule, mode + wins over mode - since there are more + entries than - entries. Note that the preference policy is not applicable to this case since the conflict is resolved before this rule is triggered; however in other hierarchies the conflict need not yet have been resolved.
For each strategy instance in Table 2, we use a bold font to show which policy has determined the effective authorization when applied to our example. For example, in the last strategy instance, MGP\(^{-}\), by applying the first policy (Majority), the positive authorization wins since there are two +’s (rows 4 and 5) as opposed to only one – (row 1) in Table 1. Therefore, the localization and preference policies of the MGP\(^{-}\) instance are not applicable to this case.

3.1 Algorithm Resolve()

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

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

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

In Line 4, if the majority policy should be applied before the locality policy, we count the number of positive authorizations and negative authorizations that exist in relation \(allRights;\) however, as stated in Line 5, if the majority policy should be applied after the locality policy, we first apply the locality on relation \(allRights,\) and then count the number of positive and negative authorizations. In either of these cases (Line 6), the algorithm returns the authorization which is in majority.
Algorithm Resolve(s, o, r, dRule, lRule, mRule, pRule)
- To check whether subject s has a positive or a negative authorization for right r on object o
- Default rule dRule ∈ {“+”, “-”, “0”}
- Locality rule lRule ∈ {max(), min(), identity()}
- Majority rule mRule ∈ {“before”, “after”, “skip”}
- Preference rule pRule ∈ {“+”, “-”}
- SDAG (subject hierarchy) and EACM (explicit access control matrix) are globally defined.

Output: either “+” or “-”

1. allRights ← Propagate (s, o, r, SDAG, EACM);
2. if dRule = “0” allRights ← $\sigma_{mode="d"}$ allRights;
3. else update allRights set mode=dRule where mode="d";
4. if mRule = “before” {c1 ← $\Pi_{count(mode="+" \text{allRights)}}$; c2 ← $\Pi_{count(mode="-" \text{allRights})}$}
5. if mRule = “after” {c1 ← $\Pi_{count(mode="+", dis=Rule(dis) \text{allRights})}$; c2 ← $\Pi_{count(mode="-", dis=Rule(dis) \text{allRights})}$}
6. if mRule <> “skip” {if c1 > c2 return “+”; if c2 > c1 return “-”; }
7. Auth ← $\Pi_{\text{mode=dis=Rule(dis) \text{allRights)}}$;
8. if count(Auth) = 1 return Auth;
9. return pRule;

Fig. 4. Algorithm Resolve ()

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

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

Table 3 illustrates the result of Algorithm Resolve() applied to our motivating example for several illustrative strategies. In particular, we trace the algorithm for eight strategy instances (selected from Table 2) namely D+LMP+, DGM+, DMP+, D-LP+, D+GP+, P, GMP, and MGP. Table 3 shows values of c1, c2, Auth, and the effective mode derived by the algorithm, as well as its corresponding return line number. In the table, n/a means that the algorithm does not use the corresponding variable for the conflict resolution.
For instance, if one chooses the strategy instance D’GMP’, all default values of relation allRights are replaced with “-“ (Line 3). Since the global mode of the locality policy is in place and there are one positive and one negative authorization from distance 3 in Table 1, both $c_1$ and $c_2$’s values are assigned the value 1 (Lines 5). Then, since neither positive nor negative is in majority, the algorithm continues to Line 7, and Auth is assigned the value {+,−}. Finally, since there is a conflict in Auth, Line 9 of the algorithm returns the value of preference policy, which is “-“ (indicated by P’ in the strategy instance), as the final derived decision with respect to triple <User, obj, read>.

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

<table>
<thead>
<tr>
<th>Strategy</th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>Auth</th>
<th>mode</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>D’LMP’</td>
<td>2</td>
<td>1</td>
<td>n/a</td>
<td>+</td>
<td>6</td>
</tr>
<tr>
<td>D GMP’</td>
<td>1</td>
<td>1</td>
<td>+,-</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>D MP’</td>
<td>2</td>
<td>4</td>
<td>n/a</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>D’ LP’</td>
<td>n/a</td>
<td>n/a</td>
<td>-,-,+</td>
<td>+</td>
<td>9</td>
</tr>
<tr>
<td>D’GP’</td>
<td>n/a</td>
<td>n/a</td>
<td>+</td>
<td>+</td>
<td>8</td>
</tr>
<tr>
<td>GMP’</td>
<td>1</td>
<td>0</td>
<td>n/a</td>
<td>+</td>
<td>6</td>
</tr>
<tr>
<td>P’</td>
<td>n/a</td>
<td>n/a</td>
<td>-,-,+</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>MGP’</td>
<td>1</td>
<td>0</td>
<td>n/a</td>
<td>+</td>
<td>6</td>
</tr>
</tbody>
</table>

3.2 Function Propagate()

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

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

- To obtain all authorizations, with respect to triple `<s, o, r>` by propagating explicit authorizations in `EACM` through subject hierarchy `SDAG`
- `EACM` has attributes `<subject, object, right, mode>`
- `SDAG` has attributes `<subject, child>`
- `ancestors(s) = \{s\} \cup \{x \exists y <y,s> \in SDAG \wedge x \in ancestors(y)\}`

Output: table `allRights`

1. `SDAG' \leftarrow \sigma_{subject \in ancestors(s), child \in ancestors(s)} SDAG;`
2. `i = 0;`
3. `P \leftarrow \Pi_{subject, object, permission, i, mode} (SDAG' \Join_{permission=r, object=o} \sigma_{EACM});`
4. `Roots \leftarrow \Pi_{subject} SDAG' - \Pi_{child} SDAG' - \Pi_{subject} P;`
5. `P \leftarrow P \cup Roots \times \{<o, r, i, \text{“d”}>\};`
6. `P' \leftarrow \sigma_{subject \neq s} P;`
7. `i = i + 1;`
8. `P' \leftarrow \Pi_{child, object, permission, i, mode} P' \Join_{subject \neq s} SDAG';`
9. `P \leftarrow P \cup P';`
10. `P' \leftarrow \sigma_{subject \neq s} P;`
11. `until P' = \emptyset ;`
12. `return \sigma_{subject = s} P;`

Fig. 5. Function `Propagate()`

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

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

In Line 4, we store all unlabelled root subjects of `SDAG'` into a relation called `Roots`. For instance, `Roots` contains `{S_1, S_6}` if applied to our motivating example. In...
Line 5, for each root subject with no explicit authorization with respect to $o$ and $r$, we insert an additional row into relation $P$ to assign it the default authorization. This results in the third and fourth entries in Table 4. In Line 6, we select as $P'$ all identified authorizations other than those on the sink node $s$.

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

<table>
<thead>
<tr>
<th>subject</th>
<th>object</th>
<th>right</th>
<th>dis</th>
<th>mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_2$</td>
<td>obj</td>
<td>read</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>$S_5$</td>
<td>obj</td>
<td>read</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>$S_1$</td>
<td>obj</td>
<td>read</td>
<td>0</td>
<td>d</td>
</tr>
<tr>
<td>$S_6$</td>
<td>obj</td>
<td>read</td>
<td>0</td>
<td>d</td>
</tr>
<tr>
<td>User</td>
<td>obj</td>
<td>read</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>$S_7$</td>
<td>obj</td>
<td>read</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>User</td>
<td>obj</td>
<td>read</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>$S_3$</td>
<td>obj</td>
<td>read</td>
<td>1</td>
<td>d</td>
</tr>
<tr>
<td>User</td>
<td>obj</td>
<td>read</td>
<td>1</td>
<td>d</td>
</tr>
<tr>
<td>$S_5$</td>
<td>obj</td>
<td>read</td>
<td>1</td>
<td>d</td>
</tr>
<tr>
<td>$S_8$</td>
<td>obj</td>
<td>read</td>
<td>2</td>
<td>+</td>
</tr>
<tr>
<td>$S_5$</td>
<td>obj</td>
<td>read</td>
<td>2</td>
<td>d</td>
</tr>
<tr>
<td>User</td>
<td>obj</td>
<td>read</td>
<td>2</td>
<td>d</td>
</tr>
<tr>
<td>User</td>
<td>obj</td>
<td>read</td>
<td>3</td>
<td>+</td>
</tr>
<tr>
<td>User</td>
<td>obj</td>
<td>read</td>
<td>3</td>
<td>d</td>
</tr>
</tbody>
</table>

### 3.3 Computational Analysis

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

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