Protecting Information Sharing in Distributed Collaborative Environment

PH.D DISSERTATION

BY

LI, MIN

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF SOUTHERN QUEENSLAND IN FULLFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

PRINCIPAL SUPERVISOR: DR. HUA WANG
ASSOCIATE SUPERVISOR: DR. ASHLEY PLANK

SEPTEMBER, 2010

DEDICATION

Dedicated to my parents Zicheng Li and Xinhua Jian

and

my husband Xiaoxun Sun

STATEMENT

I hereby declare that the work presented in this dissertation is in my own and is, to the best of my knowledge and belief, original except as acknowledgement in the text. It has not previously been submitted either in whole or in part for a degree at this or any other university.

Min Li

Signature of Candidate Date

ENDORSEMENT

Signature of Supervisor Date

ACKNOWLEDGEMENT

Research is never conducted in isolation. Rather, numerous people have contributed to this thesis through their ideas, discussions, feedback, and support.

First and foremost, I would like to thank my advisor Dr. Hua Wang. His guidance and insight have shaped me as a researcher, and I am extremely fortunate to have had the opportunity to work with him over the past few years. In addition, I would like to thank Dr. Ashley Plank for his support, feedback, and encouragement.

I also owe a great deal of gratitude to the Centre for Systems Biology (CSBi), Department of Mathematics & Computing, Faculty of Science and Research and Higher Degree office of University of Southern Queensland for providing the excellent study environment and financial support. It is a great pleasure to study at the Department of Mathematics & Computing. I also acknowledge Mr. Greg Otto for his help on proof-reading the dissertation.

Finally, on a personal note, I would never have made it this far without the continued support and encouragement of my family: Zicheng Li, Xinhua Jian, Gang Li, and Xiaoxun Sun.

ABSTRACT

This thesis focuses on three aspects (i.e., role-based access control, role-based delegation and privacy-aware access control) of developing a systematic methodology for information sharing in distributed collaborative environments. We develop techniques for setting up secure group communication and providing accesses to group members for many database systems, which incorporate new security constrains and policies raised by current information technologies. We create new forms of access control models to identify and address issues of sharing information in collaborative environments and to specify and enforce privacy protection rules to support identified issues.

In role based access control systems (RBAC) permissions are associated with roles, and users are made members of appropriate roles thereby acquiring the roles' permissions. This greatly simplifies management of permissions. Roles are created for various job functions in an organization and users are assigned roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. Roles can be granted new permissions as new applications and systems are incorporated, and permissions can be revoked from roles as needed. The principal motivation of RBAC is to simplify administration. In large organizations the number of roles can be in the hundreds or thousands, and users can be in the tens or hundreds of thousands, maybe even millions. Effective management of permission-role assignment could be very useful in practice to avoid the security breach, especially when conflicting permissions granted to the same role. Constraints are an important aspect of RBAC and are a powerful mechanism for laying out higher level organizational policy. Even for the usage control (UCON) model, constraints are discussed less and no formal language is proposed to describe constraints precisely. An appealing is to study constraints formally in RBAC and UCON models. Our work looks at proposing formal approaches to

check conflicts and help allocate permissions without compromising security in RBAC and proposing a formal language to specify constraints for system designers and administrators in UCON models.

Delegation requirement arises when a user needs to act on another's behalf to access resources. Essentially, in a multi-agent system, delegation becomes the primary mechanism of inter-agent collaboration and cooperation. However, the previous delegation model could not work efficiently in large systems and perform the sensitive delegation task within the broad area of security. In this thesis, we introduce a flexible ability-based delegation model within RBAC. Moreover, to avoid risk during the delegation process, we propose a secure multi-level delegation model, where a projection between the reliability of delegatees and the sensitivity of delegated tasks is built. Our multi-level delegation model allows that a delegatee in a higher trust level can be assigned with a higher level task.

With the widespread use of information technology, privacy protection becomes a major concern and it could not be easily achieved by traditional access control models. In this thesis, we propose a privacy-aware access control model with generalization boundaries, which could maximize data usability while, minimizing disclosure of privacy. Moreover, our privacy-aware access control model provides a much finer level of control. Although Hippocratic database enforced the fine-grained disclosure policy through creating a privacy authorization table, but it does not allow to distinguish which particular method is used for fulfilling a service in a real world case. We use a goal-oriented approach to analyze privacy policies of the enterprises involved in a business process, in which one can determine the minimum disclosure of data for fulfilling the root purpose with respect to customer's maximum trust. We provide efficient algorithms to automatically derive the optimal way of authorizations needed to achieve a service from enterprise privacy policies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Collaborative systems, groupware, or multi-user applications allow groups of users to communicate and cooperate on common tasks. Example systems include a wide range of applications such as audio/video conferencing, collaborative document sharing/editing, distance learning, workflow management systems, and so on. All of these systems contain information and resources with different degrees of sensitivity. The applications deployed in such systems create, manipulate, and provide access to a variety of protected information and resources. Balancing the competing goals of collaboration and security is difficult because interaction in collaborative systems is targeted towards making people, all who need it, whereas information security seeks to ensure the availability, confidentiality, and integrity of these elements while providing it only to those with proper authorization. Protection of contextual information and resources in such systems therefore entails addressing several requirements not raised by traditional single-user environments, due in part to the unpredictability of users and the unexpected manners in which users and applications interact in collaborative sessions. Among the several areas of security under consideration for collaborative environments, authorization or access control is particularly important.

Role-based access control (RBAC) is a technology that is attracting a great deal of attention, particularly for commercial applications, because of its potential for reducing the complexity and cost of security administration in large networked applications. Over the past decade, interest in RBAC has increased dramatically. In the late 1980s and early 1990s researchers began recognizing the virtues of roles as an abstraction for managing privileges within applications and database management systems. A role was seen as a job or position

within an organization. A role exists as a structure separate from that of the users who were assigned to the roles. Dobson and McDermid [33] used the term functional roles. Baldwin [10] called these structures named protection domains (NPDs) and stated that they could be related and organized into hierarchies based on NPD permission subsets. Also recognized was the use of roles in support of the principle of least privilege in which a role is created with minimum permissions in specification of duty requirements [95]. The Brewer and Nash model [12] presented a basic theory for use in implementing dynamically changing access permissions. The model is described in terms of a particular commercial security policy, known as the Chinese wall. The model is developed by first defining what a Chinese wall means and then defining a set of rules (SoD requirements) such that no user can ever access data from the wrong side of the wall. Nash and Poland [72] discussed the application of role-based security to cryptographic authentication devices commonly used in the banking industry.

Bell and LaPadula [11] formalized military access control rules into a mathematical model suitable for defining and evaluating computer security systems. As formulated in this model, multilevel secure systems implement the familiar government document classification rule: Users are only allowed to access information that is classified at or below their own clearance level. Conceptually, this is a very simple policy, readily understood and followed by humans. In 1994, Nyanchama and Osborn [73] proposed a very generalized form of role organization called a role graph model. The authors showed that roles could be organized based on three role relationships: partial, shared, and augmented privileges. The role graph model is particularly useful in analyzing privilege sharing, which is critical in detecting and preventing conflict of interest relationships between roles. In 1996, Sandhu and colleagues [84] introduced a framework of RBAC models, called RBAC96. Since then, the concept of RBAC becomes a widely deployed and highly successful alternative to conventional discre-

tionary and mandatory access controls [1, 35, 36]. The principal idea in RBAC is that users and permissions are assigned to roles. Users acquire permissions indirectly via roles. This remarkably simple idea has many benefits and elaborations. Administration of authorization is much simplified in RBAC. Separation of user-role assignment and permission-role assignment facilitates different business processes and administrators for these tasks. Modifications to the permissions of a role immediately apply to all users in the role. Users are easily deassigned and assigned roles as their job functions and responsibilities change. There are two major elaborations of the simple RBAC concept. One elaboration is to have hierarchical roles, and another elaboration is to have separation of duty and other constraints.

In RBAC, a user is a human being or an autonomous agent, a role is a collection of permissions needed to perform a certain job function within an organization, a permission is an access mode that can be exercised on objects in the system, and a session relates a user to possibly many roles. When a user logs in the system he establishes a session and, during this session, he can request activation of some of the roles he is authorized to play. An activation request is granted only if the corresponding role is enabled at the time of the request and the user requesting the activation is entitled to activate the role at the time of the request. If an activation request is satisfied, the user issuing the request obtains all the permissions associated with the role he has requested to activate. On the sets Users, Roles, Permissions, and Sessions, several functions are defined. The user assignment (UA) and the permission assignment (PA) functions model the assignment of users to roles and the assignment of permissions to roles, respectively. A user can be authorized to play many roles, and many users can be authorized to play the same role. Moreover, a role can have many permissions, and the same permissions can be associated with many roles. The user function maps each session to a single user, whereas the function role establishes a mapping between a session and a set of roles (i.e., the roles that are activated by the corresponding user in that session).

On Roles, a hierarchy is defined, denoted by $\geq$. If $r_i \geq r_j$, $r_i, r_j \in$ Roles, then role r_i inherits the permissions of role r_j . The RBAC model consists of the following components.

- sets Users , Roles , Permissions , and Sessions , representing the set of users, roles, permissions, and sessions, respectively;
- PA: Roles$\longrightarrow$Permissions, the permission assignment function, that assigns to roles the permissions needed to complete their jobs;
- UA: Users$\longrightarrow$Roles, the user assignment function, that assigns users to roles;
- user: Sessions$\longrightarrow$ Users, that assigns each session to a single user;
- role: Sessions $\longrightarrow 2^{Roles}$, that assigns each session to a set of roles; and
- $RH \subseteq$ Roles $\times$ Roles, a partially ordered role hierarchy (written $\geq$).

RBAC is policy-neutral and flexible. The policy that is enforced is a consequence of the detailed configuration of various RBAC components. RBAC's flexibility allows a wide range of policies to be implemented. Examples of this flexibility to support different policies can be found in [34, 86, 104]. In access control systems, the number of roles and users and permissions associated to roles in a large enterprise system can be hundreds or thousands. Managing these roles, users, permissions and their interrelationships is a vital challenge that is often highly decentralized and delegated to a small team of project groups. RBAC allows us to model security from the perspective of the organization, because we can align security modeling to the roles and responsibilities in the organization. Most large organizations have some business rules related to access control policy such as need-to-know, separation of duty, rotation of sensitive job positions, and so on. Delegation of authority is an important method to implement the rules. The delegation requirement arises when a user needs to act on another user's behalf to access resources. The basic idea behind delegation is that

some active entity in a system delegates authority to another active entity to carry out some functions on behalf of the former.

A number of models dealing with various aspects of delegation have been published [14, 118, 36, 76]. RBDM0 [7, 8] is the first delegation model based on role. RBDM deals with a flat and hierarchical role, multi-step delegation, in particular, which is user-to-user delegation primarily based on roles. L. Zhang *et al* [117] presented a rule-based delegation model called RDM2000. Their model supports the specification of delegation authorization rules to impose restrictions on which roles can be delegated to whom. X. Zhang *et al* [114] proposed a permission-based delegation model called PBDM, which supports both role and permission level delegation. Their model controls delegation operations through the notion of delegatable roles such that only permissions assigned to these roles can be delegated to others. Furthermore, Wang *et al*. [100] proposed a role-based delegation model which support user-to-group delegation. In [30], Crampton and Khambhammettu proposed a delegation model that supports both grant and transfer. Atluri andWarner [6] studied how to support delegation in workflow systems. They extended the notion of delegation to allow conditional delegation, where conditions can be based on time, workload and task attributes. One may specify rules to determine under what condition a delegation operation should be performed. Role-based delegation based on role-based access control (RBAC) has proven to be a flexible and useful access control for information sharing in a distributed collaborative environment [102].

In recent years, there have been several attempts to extend access-control models beyond the basic access matrix model of Lampson, which has dominated this area for over three decades. The concept of UCON was recently introduced in the literature by Park and Sandhu as a fundamental enhancement of the access matrix [83]. The UCON model provides a comprehensive framework for the next-generation access control. A UCON system

consists of six components: subjects and their attributes, objects and their attributes, rights, authorizations, obligations, and conditions. The authorizations, obligations, and conditions are the components of the UCON decisions. UCON extends the traditional access-control models in one aspect, in that the control decision depends not only on authorizations, but also on obligations and conditions. Obligations are activities that are performed by the subjects or by the system. For example, playing a licensed music file requires a user to click an advertisement and register in the authors web page. Such an action can be required before or during the playing process. Conditions are system and environmental restrictions that are not directly related to subject or object attributes, such as the system clock, the location, system load, system mode, etc. Another way in which UCON extends traditional access control models is the concepts of continuity and mutability. A complete usage process consists of three phases through time: before usage, ongoing usage, and after usage. The control-decision components are checked and enforced in the first two phases, named pre-decisions and ongoing decisions, respectively. The presence of ongoing decisions is called continuity. Mutability means that the subject or object attribute value may be updated to a new value as a result of accessing. Along with the three phases, there are three kinds of updates: pre-updates, ongoing updates, and post-updates. All these updates are performed and monitored by the security system as the access is being attempted by the subject to the object. Changing subject and object attributes has an impact on other ongoing or future usage of permissions involving this subject or object. This aspect of mutability makes UCON very powerful. The new expressive power brought in by UCON is very germane to the automated and seamless security administration required in environments.

As the internet becomes one of the most important infrastructures for modern societies, systems need efficient access control policies in order to effectively protect system security and privacy. This reflects the growing attention of customers to their personal information

and the increasing number of laws, policies, and regulations that are intended to safeguard it. By demonstrating good privacy practices, many enterprises try to utilize information analysis and knowledge extraction to provide better services to individuals without violating individual privacy. As privacy becomes a major concern for both consumers and enterprises, much research effort has been devoted to the development of privacy protecting technology [2, 3, 5, 56]. As an important step for helping users to gain control over the use of their personal information, the W3C has proposed the Platform for Privacy Preferences (P3P) [121]. P3P allows websites to encode their privacy practice, such as what information is collected, who can access the data for what purposes, and how long the data will be stored by the sites, in a machine-readable format. Even though P3P provides a standard means for enterprises to make privacy promises to their users, P3P does not provide any mechanism to ensure that these promises are consistent with the internal data processing.

The Enterprise Privacy Authorization Language (EPAL) [119] proposed by IBM is a formal language for writing enterprise privacy policies to govern data handling practices in IT systems. An EPAL policy defines lists of hierarchies of data-categories, user-categories, and purposes. User-categories are the entities (users/groups) that use collected data, and data-categories define different categories of collected data that are handled differently from a privacy perspective. Purposes model the services for which data is intended to be used. An EPAL policy also defines sets of actions, obligations, and conditions. Actions model how the data is used, and obligations define actions that must be taken by the environment of EPAL. Lastly, conditions are boolean expressions that evaluate the context. Privacy authorization rules are defined using these elements, and each rule allows or denies actions on data-categories by user-categories for certain purposes under certain conditions while mandating certain obligations.

The concept of Hippocratic databases, incorporating privacy protection within relational

database systems, was introduced by Agrawal et al. [3]. The proposed architecture uses privacy metadata, which consist of privacy policies and privacy authorizations stored in two tables. A privacy policy defines for each attribute of a table the usage purpose(s), the external-recipients and retention period, while privacy authorization defines which purposes each user is authorized to use. Recently, Lefevre et al. [56] presented an approach to enforcing privacy policy in database environments. Their work focuses on ensuring limited data disclosure, based on the premise that data providers have control over who is allowed to see their personal data and for what purpose. Rabitti et al. [80] developed a comprehensive authorization model designed for next-generation database systems. The data models considered in their work support object-oriented concepts and incorporate some key semantic data modeling concepts such as composite objects and versions. Furthermore, they formalize and develop clear semantics for various types of authorizations such as strong/weak and negative/positive authorizations. A number of key issues that arise in implementing such a model are also discussed in their work.

In this thesis we would like to explore how to invent new forms of access control models to enhance the privacy protection aspect of current information technology.

1.1 Problem Statement

RBAC has many components, thereby making RBAC administration multi-faceted. In particular we can separate issues of assigning users to roles, assigning permissions to roles, and assigning roles to roles to define a role hierarchy. These activities are all required to bring users and permissions together. In large enterprise-wide systems the number of roles can be in the hundreds or thousands, and users can be in the tens or hundreds of thousands, maybe even millions. Managing these roles and users, and their interrelationships is a formidable task. It is very difficult to maintain consistency because it may change the authorization

level, or imply high-level confidential information when more than one permission is requested and granted. Specifically, conflicts arise when assigning permissions to roles with different memberships. We believe that substantial advances in RBAC can be further made by reconsidering the foundational principles.

In a multi-agent system, delegation is the primary mechanism of inter-agent collaboration and cooperation. The basic idea behind delegation is that some active entity in a system delegates authority to another active entity to carry out some functions on behalf of the former. Essentially, a delegation operation could temporarily change the access control state so as to allow an agent to use another agent's access privileges. Due to its effect on the access control state, delegation may lead to a violation of security policies. More precisely, information breaching may happen even during the delegation phase. In an open environment, the entities are customarily alien to each other. When entering into a delegation, the delegator is entering into an uncertain interaction in which there is a risk of failure due to the delegation decisions. We are interested in developing a secure delegation model that can be recognized as a very useful component of access control systems.

Usage control (UCON) model is considered as the next generation access control model. In UCON, authorizations, obligations and conditions are decision factors employed by the usage decision functions to determine wether a subject should be allowed to access an object with a particular right. In addition to these factors, modern information system also includes another two important properties called "continuity" and "mutability". Constraints as one of the most important components have involved in the principle motivations of usage analysis and design, however, there is not much work addressing this issue. It is necessary to provide a formal language to precisely describe constraints of UCON.

While current information technology enables people to carry out their business virtually at any time in any place, privacy issues are exacerbated when personal information is

collected, stored and used in various information systems. Privacy protection has become a critical issue in the development of information systems. We emphasize that privacy protection cannot be easily achieved by traditional access control models. The first reason is that while traditional access control models focus on which user is performing which action on which data object, privacy policies are concerned with which data object is used for which purpose(s). Another difficulty of privacy protection is that the comfort level of data usage varies from individual to individual. We believe that the availability of new generation access control mechanisms is an important requirement of a comprehensive solution to privacy.

1.2 METHODOLOGY

Permission-role assignment is an important issue in role-based access control (RBAC). There are two types of problems that may arise in permission-role assignment. One is related to the authorization granting process in that conflicting permissions may be granted to a role and, as a result, users with the role may have or derive a high level of authority. The other is related to authorization revocation. When a permission is revoked from a role, the role may still have the permission from other roles. To solve these problems, we discuss granting and revocation models related to mobile and immobile memberships between permissions and roles, and propose authorization granting algorithms to check conflicts and help allocate the permissions without compromising security. Moreover, the new revocation models, local and global revocation, are well studied. The revocation algorithms based on relational algebra and operations provide a rich variety.

Delegation requirement arises when a user needs to act on another user's behalf to access resources. In todays highly dynamic distributed systems, collaboration is necessary for information sharing with others, so a user may want to delegate a collection of permissions, named an ability, to another user or all members of a group. Further, delegation is a mechanism that

allows one agent to act on another's privilege. It is important that the privileges should be delegated to a person who is trustworthy. Based on this fact, we build a new ability-based delegation model (ABDM) and develop its delegation algorithm. The framework include both ability-based user-user delegation and user-group delegation. Also, in order to assess how trustworthy a delegatee is, we devise trust evaluation techniques to describe a delegatee's trust history and predict the future trend of trust. We perform an in-depth study on how to manage a delegation in a secure manner and develop a secure delegation model by taking trust into account.

Constraints are an important aspect of RBAC and are a powerful mechanism for laying out higher level organizational policy. Even for the usage control (UCON) model, constraints are discussed less and no formal language is proposed to describe constraints precisely. An appealing is to study constraints formally in UCON models. Therefore, we specify constraints of the UCON model with object constraints language (OCL). With OCL, we provide a tool to precisely describe constraints for system designers and administrators. The specification also provides the precise meaning of the new features of UCON, such as the mutability of attributes and the continuity of usage control decisions.

To ease privacy concerns, many important mechanisms are proposed to guarantee the respect of privacy principles in data management. However, the issues like purpose hierarchies, task delegations and minimal privacy cost are missing from the proposed mechanisms. We extend these mechanisms in order to support inter-organizational business processes. In particular, we organize purposes into purpose directed graphs through AND/OR decompositions, which support the delegation of tasks and authorizations when a host of partners participating in the business service provides different ways to achieve the same service. Specially, customers have control of deciding how to get a service fulfilled on the basis of their personal feeling of trust for any service customization. Quantitative analysis is per-

formed to characterize privacy penalties dealing with privacy cost and customer's trust.

1.3 CONTRIBUTIONS

Information sharing on distributed collaboration usually occurs in broad, highly dynamic network-based environments, and formally accessing the resources in a secure manner poses a difficult and vital challenge. My PhD research focuses exclusively on how to specify and enforce policies for information sharing in distributed collaborative environments based on three main contributions.

The first main contribution of this research is to extend the current research in role-based access control and usage control models. We provide new authorization allocation algorithms for RBAC along with mobility that is based on relational algebra operations. The authorization granting algorithm, local and global revocation algorithm defined in this section can automatically check conflicts when granting more than one permission as mobile or immobile member to a role in the system. In the UCON model, we analyze various kinds of constraints represented with object constraint language such as decision actor constraints and mutability constraints etc. We also provide a tool to precisely describe constraints for system designers and administrators and show the flexibility and expressive capability of this model by specifying the core models of UCON with extensive examples.

The second contribution is improving the delegation models by integrating new techniques to develop efficient algorithms and build efficient delegation frameworks. We proposed a flexible ability-based delegation model by extending user to group and permission to ability. We develop delegation algorithms and analyze the delegating framework including delegating authorization and revocation with constraints on an ability-based delegation. Moreover, the multi-level delegation model proposed in this work allows that a delegatee in a higher trust level can be assigned with a higher level of task. The delegation task lev-

els are classified according to information sensitivity, while, the trust levels combine trust values and trust trend together to indicate to what extent a delegatee is reliable or trustworthy. The effectiveness of our proposed multi-level delegation model is investigated and the experimental results confirm the advantages of our model in privacy protection.

The third contribution is to propose new mechanisms in privacy protecting systems for enhancing current privacy methods with respect to balancing data privacy and usability. We propose a privacy-aware access control model with generalization boundaries, which can satisfy the requirements of both data providers and data users. Compared with traditional access models, our model provides a much finer level of control as the access control decision is based on the question of "how much information can be allowed for a certain user", rather than "is information allowed for a certain user or not". Since purpose plays an important role in order to capture the intended usage of information, we organize purposes in hierarchal manner through AND/OR decomposition and use a goal-oriented approach to analyze the privacy policies involved in a business process. We also provide efficient algorithms to determine the optimal privacy-aware path for achieving a service. This allows the automatic derive action of access control policies for an inter-organizational business process from the collection of privacy policies associated with different participating enterprises.

1.4 Organization of the Thesis

Although role-based access control is the de facto access control model for collaborative environment, not much work appears in studying the conflicts when assigning permissions to roles with different memberships. In Chapter 2, we analyzes authorization granting and revocation models with the mobility of permission-role relationships. With the increase of the shared information and resources in the collaborative environment, unauthorized access to the information by illegal users that leads to the leakage of the data also increases. For

better performance, it is important to keep resources and the information integrity from the unexpected use by unauthorized user. Therefore, delegation models were proposed as a support to the strong demand for the authentication and the access control of distributed-shared resources. In the following two chapters, we extend the traditional delegation models to meet different requirements of information sharing. Chapter 3 studies an flexible ability-based delegation model in RBAC and develops delegation algorithms. Chapter 4 illustrates multi-level delegations with trust management, where both delegation tasks and trust are organized into three levels. The proposed multi-level delegation model allows that a delegatee in a higher trust level can be assigned with a higher level of task. Technological innovations in computers and networks have enabled pervasive availability and usability of digital information bringing us opportunities for new business models and personal life styles. Because of these innovations, traditional access control models could not deal with new challenging issues for reliable and trusted controls on the usages of digital resources. The usage control model (UCON) was proposed as a comprehensive framework for the next generation access control. In Chapter 5, we specifies the usage control model with object constraint language and provides a tool to precisely describe constraints for system designers and administrators. After the Internet becomes one of the most important infrastructures for modern societies, and systems need efficient access control policy in order to effectively protect the system security and privacy. In Chapter 6, we proposes a generalization boundary technique that can satisfy the requirements of both data providers and data users. Moreover, we present a privacy-aware access control model, where the trust-based decision policy and ongoing access control policy combine together to create a secure protection system. Finally in Chapter 7, we organizes purposes in a hierarchal manner through AND/OR decomposition and introduces how to find the optimal privacy-aware path in Hippocratic databases. The conclusion of this dissertation and future research directions are discussed in Chapter 8.

CHAPTER 2

ADVANCED PERMISSION-ROLE RELATIONSHIP IN RBAC

In this chapter, we develop formal approaches to check the conflicts and therefore help allocate the permissions without compromising security. We analyze authorization granting and revocation models with the mobility of permission-role relationships. Our main contribution in this chapter is relational algebra-based authorization granting and local, global revocation algorithms. Furthermore, we include an applicable example to illustrate our algorithms. Another contribution is that our algorithms could check conflicts when granting more than one permission as a mobile or immobile member to a role in the system. As far as we know, there is no previous work addressing these issues for permission allocation and conflict detection concerning mobile memberships.

The information in this chapter is based on a published paper [62].

2.1 INTRODUCTION

Role-based access control (RBAC) is a flexible and policy-neutral access control technology and is a promising access control technology for the modern computing environment [14, 35, 45, 118]. In RBAC, permissions(each permission is a pair of objects and operations) are associated with roles and users are assigned to appropriate roles thereby acquiring the role's permissions. As such, a user in RBAC is a human being. It can be easily reassigned from one role to another. A role is a job function or job title and created for various job functions in an organization and users are assigned roles based on responsibilities and qualifications.

A permission is an approval of a particular mode of access to one or more objects. The relationships between users and roles, and between roles and permissions are many-to-many (i.e, a permission can be associated with one or more roles, and a role can be associated with one or more permissions) as depicted in Figure 2.1. Roles can be granted new permissions as new applications come on line and permissions can be revoked from roles as needed. Within RBAC, users are not directly granted permission to perform operations on an individual object, but permissions are associated with roles.

The RBAC model supports the specification of several aspects:

a. User/role associations $--$ the constraints specifying user authorization to perform roles;

b. Role hierarchies $--$ the constraints specifying which role may inherit all of the permissions of another role;

c. Duty separation constraints $--$ there are user/role associations indicating conflict of interest;

c1. Static separated duty (SSD) $--$ a constraint specifying that a user cannot be authorized for two different roles;

c2. Dynamic separation duty (DSD) $--$ a constraint specifying that a user can be authorized for two different roles but cannot act simultaneously in both;

d. Cardinality $--$ the maximum number of users allowed; i.e. how many users can be authorized for any particular role (role cardinality); e.g., only one manager.

Significant developments have been made within RBAC. The NIST model of RBAC [34] and Web implementation of RBAC incorporates an administrative tool that provides rudimentary support for an RBAC database that stores information about user and permission role assignments and role hierarchies [36]. Nyanchama and Osborn [76] define a role graph model that rigorously specifies operational semantics for manipulating role relations

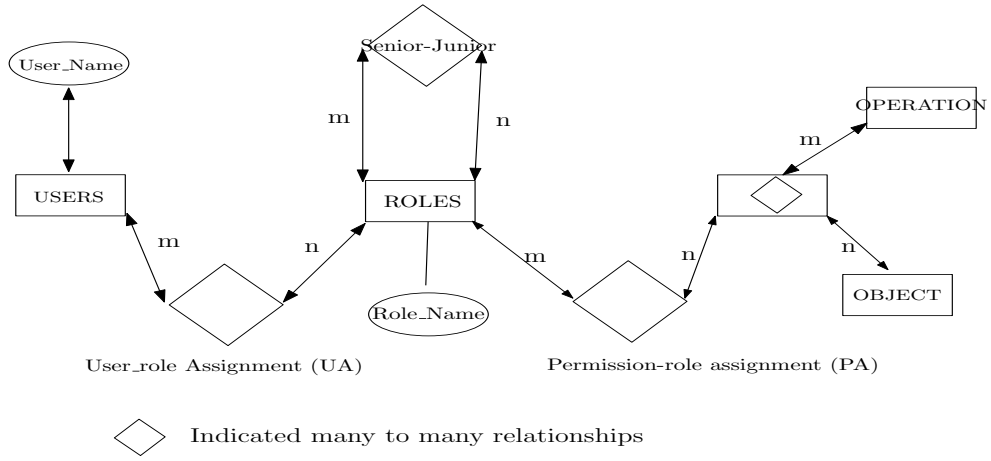


Figure 2.1: RBAC relationship

in the contexts of a role hierarchy. ARBAC97 builds on these previous attempts to construct administrative models [87] over all aspects of the RBAC model. Sandhu and Munawer [88] extends the ARBAC97 model by adding the concept of mobile and immobile permissions for the first time in this area. In [88], the authors distinguished two kinds of membership in a role. Immobile membership grants the role to have the permission, but does not make that permission eligible for further role assignment. Mobile membership on the other hand, covers both aspects.

However, there is a consistency problem when using RBAC management. For instance, if there are hundreds of permissions and thousands of roles in a system, it is very difficult to maintain consistency because it may change the authorization level, or imply high-level confidential information when more then one permission is requested and granted. Specifically, [88] does not mention conflicts when assigning permissions to roles. Therefore, there is no support to deal administrative role with regular roles, especially mobile and immobile members.

The organization of this chapter is as follows. In Section 2.2, we consider the mobility of permission-role relationship and problems related to permission assignment and revocation.

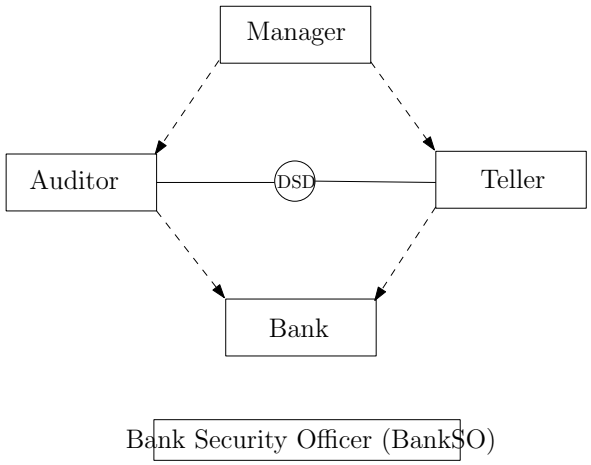


Figure 2.2: Administrative role and role relationships in a bank

The relational algebra-based authorization granting and revocation algorithms are given in Section 2.3. In Section 2.4, we review an anonymity scalable electronic commerce payment scheme and apply algorithms to this scheme. Comparisons with previous work are discussed in Section 2.5. Finally, we summary this chapter in Section 2.6.

2.2 MOTIVATION AND PROBLEM DEFINITIONS

There are two kinds of membership between permissions and roles, namely mobile and immobile [88]. Immobile membership grants the role the permission but does not make that permission eligible for further role assignments. Mobile membership on the other hand covers both aspects which means the role has the permission and the permission also becomes eligible for assignment to other roles by appropriate administrative roles.

The distinction between mobile and immobile membership can be very important in practice. Figure 2.2 shows the administrative and regular roles that exist in a bank department. The permission-role assignment allows us to give BankSo the authority to take a permission assigned to Manager and grant it to roles Teller, Auditor, and Bank. The idea is that each

administrative role can delegate permissions of the senior role to more junior roles. While this may be acceptable for most permissions of a senior role, it is likely that some permissions are not suitable for such delegation. For instance in Figure 2.2, suppose 'approving a loan' is a permission of the role Manger, which should only be executed by the Manger. Consider the two kinds of membership between permissions and roles, if this permission is assigned to the role Manger as a mobile member, it is possible that all the roles junior to the Manger can hold this permission through permission-role assignment, which leads to security breach. So this permission can only be assigned to Manager as immobile while the others can be assigned as mobile.

This example demonstrates that situations with mobile and immobile relationship between permissions and roles can be very useful in practice to avoid the security breach. Throughout the paper, we consider the following two problems that may arise in permission-role assignment.

Authorization granting problem: Is a permission in conflict with the permissions of a role when granting the permission to the role as a mobile or immobile member?

Authorization revocation problem: Has a permission with mobile or immobile membership of a role been revoked from the role?

Conflicting permissions may be granted to a role in permission-role assignment. For example, the permission for approving a loan in a bank and that of funding a loan are conflicting. These two permissions can not be assigned to a role at the same time. It is easy to find conflicts between permissions when assigning permissions to a role in a small database but it is hard to find them when there are thousands of permissions in a system. Moreover, it is even more complicated if taking mobile and immobile permissions into account. Our aim is to provide relational algebra algorithms to solve these problems and then automatically check conflicts when assigning and revoking.

PermName	Oper	Object	ConfPerm
Approval	approve	cash / check	Funding
Funding	invest	cash	Approval
Audit	audit	record	Teller
Teller	transfer	cash	Audit

Table 2.1: Example of the relation PERM

For convenience, we recall some basic definitions in paper [104] with no further explanation. Let D be a database with a set of relations REL and a set of attributes Attri. REL includes PERM, ROLE-PERM and SEN-JUN etc. Attri includes attributes such as RoleName, PermName, Senior and Junior, etc.

PERM is a relation of PermName, Oper, Object and ConfPerm. Perm-Name is the primary key for the table and is the name of the permission in the system. Oper is the name of the operation granted. It contains information about the object to which the operation is granted. Object is the item that can be accessed by the operation, which may be a database, a table, a view, an index or a database package. ConfPerm is a set of permissions that is conflicting with the PermName in the relation. For example, a staff member in a bank cannot have permissions of approval and funding at the same time (as well as permissions of audit and teller). The relation of PERM is expressed in Table 2.1.

SEN-JUN is a relation of roles in a system. SEN and JUN are the senior and junior of the two roles, senior roles are shown at the top of the role hierarchies. Senior roles inherit permissions from junior roles. For example, Table 2.2 expresses the SEN-JUN relationship in Figure 2.2, and role 'Manager' is the senior role of role 'Teller' and inherits all permissions of 'Teller'.

ROLE-PERM is a relation between the ROLES and the PERM, listing what permissions are granted to what roles. For example, in Table 2.3, permission 'Approval' is assigned to role 'Teller' and the permission 'Funding' to role 'Manager'.

Senior	Junior
MANAGER	AUDITOR
MANAGER	TELLER
AUDITOR	BANK
TELLER	BANK

Table 2.2: SEN-JUN relation in Figure2.2

RoleName	PermName
MANAGER	Funding
TELLER	Approval

Table 2.3: Example of relation ROLE-PERM

Admin.role	Prereq.condition	Role Range
BankSO	Manager$\wedge\overline{Teller}$	[Auditor, Auditor]
BankSO	Manager$\wedge\overline{Auditor}$	[Teller, Teller]

Table 2.4: Example of *can-assignp-M* in Figure2.2

Admin.role	Prereq.condition	Role Range
BankSO	Manager	[Auditor, Audior]
BankSO	Manager	[Teller, Teller]

Table 2.5: Example of *can-assignp-IM* in Figure2.2

2.3 AUTHORIZATION GRANTING AND REVOCATION ALGORITHMS

In this section, we provide granting and revocation algorithms based on relational algebra. As discussed before, a permission's membership in a role can be mobile or immobile, so each role x is separated into two sub-roles Mx and IMx. Note that membership in Mx is mobile whereas membership in IMx is immobile.

A role x' has all permissions of a role x when $x' > x$[1]. A permission p is an explicit member of a role x if $(p, x) \in PA$ and p is an implicit member of a role x if for some role $x' < x, (p, x') \in PA$. Combining mobile and immobile membership with the notion of explicit and implicit membership gives us four distinct kinds of role membership:

(1) Explicit mobile member $EMx = \{p|(p, Mx) \in PA\}$

(2) Explicit immobile member $EIMx$=$\{p|(p, IMx) \in PA\}$

(3) Implicit mobile member $ImMx = \{p|\exists x' < x, (p, Mx') \in PA\}$

[1] $x' > x$ means role x' is senior to x; $x' < x$ means role x' is junior to x.

Admin.role	Prereq.condition	Role Range
BankSO	Bank	[Bank, Manager]

Table 2.6: $can\text{-}revokep\text{-}M$ in Figure2.2

Admin.role	Prereq.condition	Role Range
BankSO	Bank	[Bank, Bank]

Table 2.7: $can\text{-}revokep\text{-}IM$ in Figure2.2

(4) Implicit immobile member $ImIMx = \{p|\exists x' < x, (p, IMx') \in PA\}$

It is possible for a permission to have more than one kind of membership in a role at the same time. Hence there is strict precedence among these four kinds of membership [2].

$$EMx > EIMx > ImMx > ImIMx$$

A prerequisite condition is evaluated for a permission p by interpreting role x to be true if $p \in EMx \vee (p \in ImMx \wedge p \notin EIMx)$ and $\overline{x}$ to be true if $p \notin EMx \wedge p \notin EIMx \wedge p \notin ImMx \wedge p \notin ImIMx$.

For a given set of roles R, let AR be a set of administrative roles and CR denote all possible prerequisite conditions that can be formed using the roles in R. Not every administrator can assign a permission to a role. The following relations provide what permissions an administrator can assign mobile members or immobile members with prerequisite conditions.

$Can\text{-}assignp\text{-}M$, used to assign the permission as mobile members, is a relation in $AR \times CR \times 2^R$. While $can\text{-}assignp\text{-}IM$ assigns the permission as immobile members. Table 2.4 and 2.5 show the example of these two relations. The meaning of $(BankSO, Manager \wedge \overline{Teller}, [Auditor, Auditor]) \subseteq$ ***can-assignp-M*** is that $BankSO$ can assign a permission whose current membership satisfies the prerequisite condition $Manager \wedge \overline{Teller}$ to role $Auditor$ as a mobile member. $(BankSO, Manager, [Teller, Teller]) \subseteq$ ***can-assignp-IM***

[2] Even though a role can have multiple kinds of membership in a permission, at any time only one of those is actually in effect.

means that $BankSO$ can assign a permission whose current membership satisfies the prerequisite condition $Manager$ to role $Teller$ as an immobile member. To identify a role range within the role hierarchy, the following closed and open interval notation is used:

$[x, y]$=$\{r \in R | x \geq r \wedge r \geq y\}$, $(x, y]$=$\{r \in R | x > r \wedge r \geq y\}$

$[x, y)$=$\{r \in R | x \geq r \wedge r > y\}$, $(x, y) = \{r \in R | x > r \wedge r > y\}$

Suppose an administrator role (ADrole) wants to assign a permission p_j to a role r with a set of permissions P which may include mobile and immobile members. The permission p_j may be assigned as a mobile and immobile member if there is no conflict between p_j and the permissions in P. We analyze both mobile and immobile members in the following algorithm, which deals with whether the ADrole can assign the permission p_j to r with no conflicts. In algorithm 1, P^* is the extension of P, which includes the explicit and implicit members of P; i.e., $P^* = \{p | p \in P\} \cup \{p | \forall r' < r, (p, r') \in PA\}$.

Algorithm 1 provides a way to check whether or not a permission can be assigned as mobile or immobile member to a role. It can prevent conflicts when assigning a permission to a role with mobile or immobile memberships as well. After considering the authorization, we consider the revocation of permission-role membership.

In the revocation model, a prerequisite condition is evaluated for a permission p by interpreting role x to be true if $p \in EMx \vee p \in EIMx \vee p \in ImMx \vee p \in ImIMx$ and $\overline{x}$ to be true if $p \notin EMx \wedge p \notin EIMx \wedge x \notin ImMx \wedge p \notin ImIMx$. Permission-role revocation of mobile and immobile memberships are authorized by the relations $can\text{-}revokep\text{-}M \subseteq AR \times CR \times 2^R$ and $can\text{-}revokep\text{-}IM \subseteq AR \times CR \times 2^R$ respectively.

$(BankSO, Manager, [Bank, Manager)) \subseteq can\text{-}revokep\text{-}M$ in Table 2.6 means that $BankSO$ can revoke the mobile membership of a permission from any role in $[Bank, Manager)$ which satisfies the revoke prerequisite condition $Bank$. Similarly, the $can\text{-}revokep\text{-}IM$ in Table 2.7 refers to revoking the immobile membership.

Algorithm 1: Authorization granting algorithm; *Grantp(ADrole, r, p_j).*

Input: ADrole, role r and a permission p_j

Output: true if ADrole can assign p_j to r with no conflicts; false otherwise

*Step 1. /*whether the ADrole can assign the permission p_j to r or not*/*

Let $S_{M1} = \pi_{prereq.condition}(\sigma_{admin.role=ADrole}(can\text{-}assignp\text{-}M))$,
$S_{IM1} = \pi_{prereq.condition}(\sigma_{admin.role=ADrole}(can\text{-}assignp\text{-}IM))$,
and $R = \pi_{Rolename}(\sigma_{Permname=p_j}(ROLE\text{-}PERM))$

Suppose p_j is a mobile member of the role r.

If $S_1 = S_{M1} \cap R \neq \emptyset$, there exists a role $r_1 \in S_1$, such that $(p_j, r_1) \in PA$ and $r_1 \in \pi_{prereq.condition}(\sigma_{admin.role=ADrole}(can\text{-}assignp\text{-}M))$

Go to *Step 2*;

Suppose p_j is a immobile member of the role r.

If $S_2 = S_{IM1} \cap R \neq \emptyset$, there exists a role $r_2 \in S_2$, such that $(p_j, r_2) \in PA$ and $r_2 \in \pi_{prereq.condition}(\sigma_{admin.role=ADrole}(can\text{-}assignp\text{-}IM))$

Go to *Step 2*;

else

return false and stop

Step 2. / whether the permission p_j is conflicting with permissions of r or not*/*

Let $ConfPermS = \pi_{ConfPerm}(\sigma_{PermName=p_j}(PERM))$

If $ConfPermS \cap P^* \neq \emptyset$, then p_j is a conflicting permission with role r

return false;

else

return true;

Before giving out our revocation algorithms, first we introduce the concept of local and global revocation [101]. Local revocation only happens to the explicit relationship between permissions and roles, while global revocation effects all other roles which are junior to the role with the revoked permission. For local revocation, the permission is revoked only if the permission is an explicit member of the role. For example in Figure 2.3, the role r_1 still has the permission p which has been locally revoked since the role is senior to role r_2 and r_3 which are associated with the permission p. Therefore, local revocation from a role has no effect when a permission is an implicit member of the role. However, global revocation requires revocation of both explicit memberships and implicit memberships. If we globally revoke permission p from the role r_1, all the relationships between the permission p and roles junior to r_1 are revoked (see Figure 2.4). Global revocation therefore has a cascading

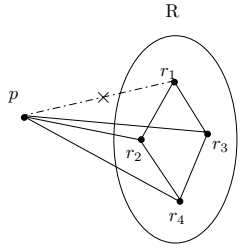


Figure 2.3: Local revocation

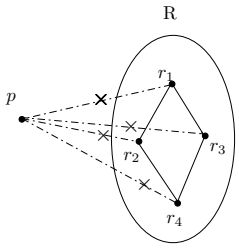


Figure 2.4: Global revocation

effect downwards in the role hierarchy. Global revocation of a permission's mobile and immobile membership from role r requires that the permission be removed not only from the explicit mobile and immobile membership in r, but also from explicit and implicit mobile and immobile membership in all roles junior to r.

Algorithms 2 and 3 are used to revoke permission $p_j \in P$ from a role r by ADrole, where P is the set of permissions which have been assigned to the role r. Algorithm 2 can be used revoke explicit mobile and immobile memberships, while Algorithm 3 can revoke explicit and implicit mobile and immobile members. It should be noted that the global revocation algorithm does not work if ADrole has no right to revoke p_j from any role in Jun.

2.4 APPLYING THE RELATIONAL ALGEBRA ALGORITHMS

In this section, we apply the new relational algebra algorithms to a consumer anonymity scalable payment scheme. We first briefly introduce the payment scheme and consider the relationships of the roles in the scheme, and then analyze applications of our relational algebra algorithms.

Algorithm 2: Local Revocation Algorithm; *Local-revoke(ADrole, r, p_j).*

Input: ADrole, role r and a permission p_j

Output: true if ADrole can locally revoke p_j from r; false otherwise

Step 1. If $p_j \notin \{p|(p,r) \in PA\}$
return false and stop.
*/*there is no effect with the operation of the local revocation since the permission P_j is not an explicit member of the role r*/*
else Go to *Step 2*. */*p_j is an explicit member of r*/*

Step 2. */*whether the ADrole can revoke the permission p_j from r or not*/*
Let $RoleRange1 = \pi_{RoleRange}(\sigma_{admin.role=ADrole}(can\text{-}revokep\text{-}M))$,
$RoleRange2 = \pi_{RoleRange}(\sigma_{admin.role=ADrole}(can\text{-}revokep\text{-}IM))$
and $Roles_{withp_j} = \pi_{RoleName}(\sigma_{PerName=p_j}(ROLE\text{-}PERM))$.
Suppose $r \in EMp_j$
If $r \in RoleRange1 \cap Roles_{withp_j} \neq \emptyset$; */*$r$ is in the role range to be revoked by ADrole in $can\text{-}revokep\text{-}M$ and the mobile membership with P_j*/*
return true;
Suppose $r \in EIMp_j$
If $r \in RoleRange2 \cap Roles_{withp_j} \neq \emptyset$; */*$r$ is in the role range to be revoked by ADrole in $can\text{-}revokep\text{-}IM$ and the immobile membership with P_j*/*
return true;
else
return false and stop. */*ADrole has no right to revoke the permission P_j from the role r*/*

2.4.1 THE ANONYMITY SCALABLE ELECTRONIC PAYMENT SCHEME

The payment scheme provides different degrees of anonymity for consumers. Consumers can decide the levels of anonymity. They can have a low level of anonymity if they want to spend coins directly after withdrawing them from the bank. Consumers can achieve a high level of anonymity through an anonymity provider (AP) agent without revealing their private information and are secure in relation to the bank because the new certificate of a coin comes from the AP agent who is not involved in the payment process.

Electronic cash has sparked wide interest among cryptographers [40, 79]. In its simplest form, an e-cash system consists of three parts (a bank, a consumer and a shop) and three main procedures (withdrawal, payment and deposit). Besides the basic participants, a third

Algorithm 3: Global Revocation Algorithm; *Global-revoke(ADrole, r, p_j).*

Input: ADrole, role r and a permission p_j
Output: true if ADrole can globally revoke p_j from r; false otherwise

Begin. If $p_j \notin P^*$
return false; /**there is no effect with the operation of the local revocation since p_j is not an explicit and implicit member of r*/*
else
(1) If $p_j \in P$ is a mobile member of the role and $r \in EMp_j$,
Local-revoke(ADrole, r, p_j); /**p_j is locally revoked as a mobile member*/*
If $p_j \in P$ is an immobile member of the role and $r \in EIMp_j$,
Local-revoke(ADrole, r, p_j); /**p_j is locally revoked as an immobile member*/*
(2) Suppose $Jun = \pi_{junior}(\sigma_{Senior=r}(SEN\text{-}JUN))$
For all $y \in Jun$ with mobile membership with the permission
Local-revoke(ADrole, y, p_j) as $y \in EMp_j$;
For all $y \in Jun$ with immobile membership with the permission
Local-revoke(ADrole, y, p_j) as $y \in EIMp_j$;
/P_j is locally revoked from all such $y \in Jun$*/*
If all local revocations are successful,
return true;
otherwise
return false.

party named anonymity provider (AP) agent is involved in the scheme. The AP agent helps the consumer to get the required anonymity but is not involved in the purchase process. The model is shown in Figure 2.5. The AP agent gives a certificate to the consumer when he/she needs a high level of anonymity.

From the viewpoint of banks, consumers can improve anonymity if they are worried about disclosure of their identities. This is a practical payment scheme for internet purchases because it has provided a solution with different anonymity requirements for consumers. However, consumers cannot get the required level of anonymity if the role BANK and AP are assigned to one user. It shows the management importance of the payment scheme. To simplify the management, we analyze its management with the relational algebra algorithms.

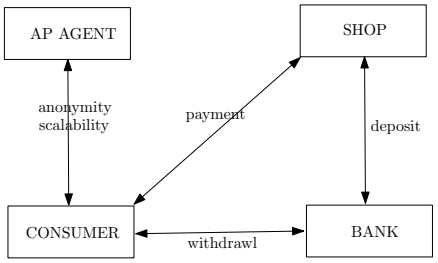


Figure 2.5: Electronic cash model

RoleName	PermName
Director(DIR)	Funding
Director(DIR)	Approval
Director(DIR)	Teller
TELLER	Approval
FPS	Approval
Bank	Teller

Table 2.9: ROLE-PERM relation

2.4.2 APPLYING THE AUTHORIZATION GRANTING ALGORITHM

Due to the length limit, we only include an application of the authorization granting algorithm. A hierarchy of roles and a hierarchy of administrative roles are show in Figure 2.6 and 2.7 respectively, we define the $can\text{-}assignp\text{-}M$ in Table 2.10. Here, we only show the process of assigning a permission to a role as a mobile member.

Here, we only analyze NSSO tuples in Table 2.10 (the analysis for APSO, BankSO and ShopSO are similar). The first tuple authorizes NSSO to assign permissions whose current membership satisfies the prerequisite condition role DIR to role M1 in the AP agent as mobile members. The second and third tuples authorize NSSO to assign permissions whose current membership satisfies the prerequisite condition role DIR to role M2 and M3 respectively as mobile members. Table 2.9 shows parts of the relations between permissions and roles in the scheme. Assume the role FPS with permission set $P = \{Approval\}$ and $P^* = P = \{Approval\}$. The administrative role NSSO wants to assign the permission $Teller$ to the role FPS as a mobile member. Using the first step of the granting algorithm ***Grantp(NSSO, FPS, Teller)***, we could get:

$S = \pi_{prereq.condition}(\sigma_{admin.role=NSSO}(\textbf{\textit{can-assignp-M}})) = \{DIR\}$ and

$R = \pi_{Rolename}(\sigma_{Permname=Teller}(\textbf{\textit{ROLE-PERM}})) = \{DIR, Bank\}$;

Since $R \bigcap S = \{DIR\} \neq \emptyset$, NSSO can assign permission $Teller$ to the role FPS as a mobile member. Applying the second step based on Table 2.1, we could get:

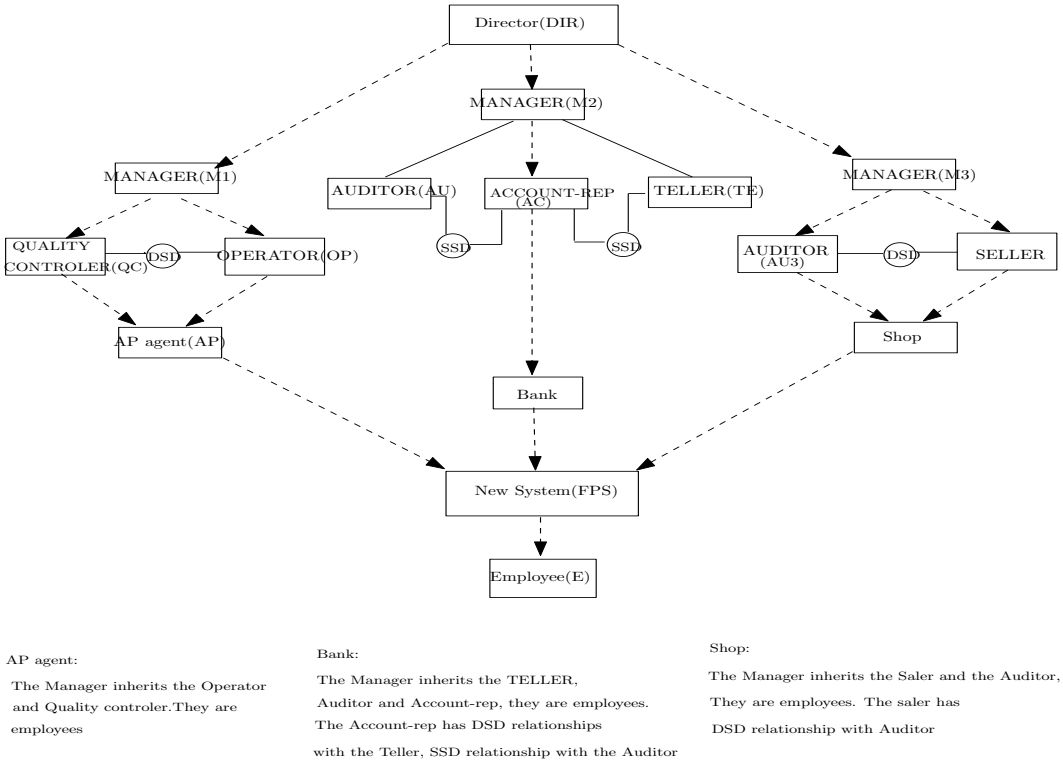


Figure 2.6: User-role assignment in the payment scheme

$ConfPermS$=$\pi_{ConfPerm}$ $(\sigma_{PermName=Teller}(PERM)) = \{Audit\}$ and

$ConfPermS \cap P^* = \emptyset$.

Hence there are no conflicts when assigning permission $Teller$ to the role FPS as a mobile member.

2.4.3 APPLICATION OF THE AUTHORIZATION REVOCATION ALGORITHM

Tables 2.11 and 2.2 give the $can\text{-}revokep\text{-}M$ and a part of senior-junior relationship of payment scheme.

Based on the Table 2.9, let us consider the permission $Approval$ is an explicit mobile member of role DIR, TELLER and FPS in the scheme. If Alice, with the activated administrative role BankSO, locally revokes $Approval$'s mobile membership from TELLER, the re-

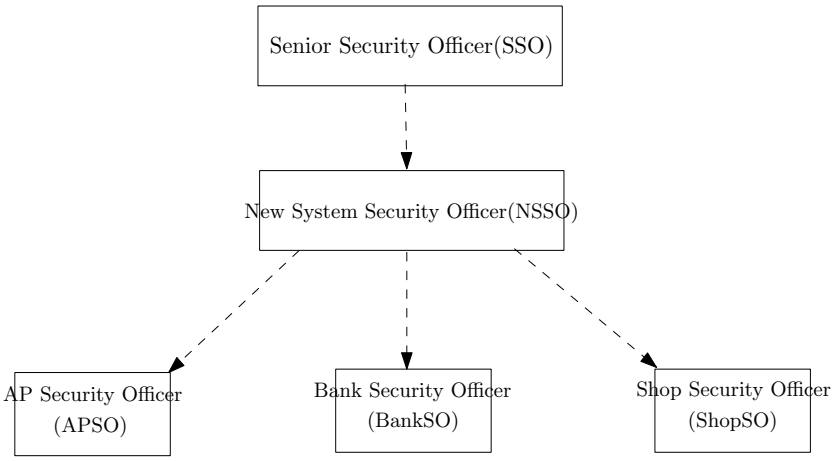


Figure 2.7: Administrative role assignment in the scheme

Admin.role	Prereq.condition	Role Range
NSSO	DIR	[M1, M1]
NSSO	DIR	[M2, M2]
NSSO	DIR	[M3, M3]
APSO	M1 $\wedge\overline{OP}$	[QC, QC]
APSO	M1 $\wedge\overline{QC}$	[OP, OP]
BankSO	M1 $\wedge\overline{TE} \wedge \overline{AU}$	[AC, AC]
BankSO	M1 $\wedge\overline{TE} \wedge \overline{AC}$	[AU, AU]
BankSO	M1 $\wedge\overline{AU} \wedge \overline{AC}$	[TE, TE]
ShopSO	M1 $\wedge\overline{SALER}$	[AUDITOR, AUDITOR]
ShopSO	M1 $\wedge\overline{AUDITOR}$	[SALER, SALER]

Table 2.10: $Can\text{-}assignp\text{-}M$ of Figure 2.6

vocation is successful by the local revocation algorithm $Local\text{-}revokep\text{-}M(BankSO, TELL\text{-}ER, Approval)$. Because

$RoleRange$=$\pi_{RoleRange}(\sigma_{admin.role=BankSO}(can\text{-}revokep\text{-}M))$=$[Bank, M2)$,

$Roles_{withApproval}$=$\pi_{RoleName}(\sigma_{PerName=Approval}\ (ROLE\text{-}PERM))$ =$\{DIR, TELLER, FPS\}$.

Therefore,

$$TELLER \in RoleRange \cap RoleswithApproval = \{TELLER\} \neq \emptyset$$

$Approval$ continues to be an implicit mobile member of TELLER since FPS is junior

Admin.role	Prereq.Condition	Role Range
NSSO	FPS	[FPS, DIR]
APSO	AP	[AP, M1]
BankSO	Bank	[Bank, M2]
ShopSO	Shop	[Shop, M3]

Table 2.11: $Can\text{-}revokep\text{-}M$

to TELLER and $Approval$ is an explicit mobile member of FPS. It is necessary to note that Alice should have enough power in the session to locally revoke $Approval$ explicitly from FPS, but she is not allowed to proceed because BankSO does not have the authority of local revocation from FPS according to the $can\text{-}revokep\text{-}M$ relation in Table 2.11. Therefore, if Alice wants to revoke $Approval$'s explicit membership as well as implicit membership from TELLER by local revocation, she needs to activate NSSO and locally revoke $Approval$ from TELLER and FPS.

If Alice, with the activated administrative role NSSO, globally revokes $Approval$'s membership from TELLER, then $Approval$ is removed not only from explicit membership in TELLER, but also from explicit (and implicit) membership in all roles junior to TELLER. Actually, using the global revocation algorithm ***Global-revokep-M(NSSO, TELLER, Approval)***, P=$\{Approval\}$=P^*. We do not need ***Local-revokep-M(NSSO, TELLER, Approval)*** since $Approval \in P$. The junior set of role TELLER is $\{$FPS$\}$. Then the permission $Approval$ has been removed from FPS as well as TELLER by running ***Local-revokep-M(NSSO, TELLER, Approval)*** and ***Local-revokep-M(NSSO, FPS, Approval)***. However, $Approval$ still has a member of DIR since it is not a junior role to TELLER based on the role hierarchy of Figure2.7.

2.5 RELATED WORK AND COMPARISONS

Our work substantially differs from [88] in two aspects. First, the paper [88] only introduce the definition of mobility of permission-role membership in permission-role assignment. By contrast, we discuss various cases in detail and focus on possible problems with mobility of permission-role relationship. Second, the authors only described the management of permission-role assignment with mobility in [88], but do not mention conflicts when assigning permissions to roles. Therefore, there is no support to deal administrative roles with regular roles in the proposal, especially mobile and immobile members. In this section, we present a number of special authorization algorithms for access control, especially the local and global revocation algorithms which have not been studied before. These algorithms provide a rich variety of options that can handle the document of administrative roles with permissions as mobile and immobile members. In our earlier work [103], we developed authorization approaches for permission-role assignment. The work in this section is an extension of that study. Actually, if all membership is restricted to being mobile, our algorithms can imply the algorithms described in [103]. Moreover, compared with [103], mobile, immobile memberships and prerequisite conditions are discussed in this paper.

2.6 SUMMARY

In this chapter, we provide new authorization allocation algorithms for RBAC along with mobility that is based on relational algebra operations. The authorization granting algorithm, local and global revocation algorithm defined in this section can automatically check conflicts when granting more than one permission as mobile or immobile member to a role in the system. We have also discussed how to use these algorithms for an electronic payment scheme.

CHAPTER 3

ABDM: AN EXTENDED FLEXIBLE DELEGATION MODEL IN RBAC

In this chapter, we extend user to group and permission to ability. Based on this, we proposed a flexible ability-based delegation model and developed delegation algorithms accordingly. We analyze the delegating framework including delegating authorization and revocation with constraints on an ability-based delegation. To our best knowledge, our introduced ability-based delegation model has not been discussed before.

The information in this chapter is based on a published paper [59, 61].

3.1 INTRODUCTION

In role based access control systems, the delegation requirement arises when a user needs to act on another user's behalf to access resources. This might be only within a limited time, for example, a vacation, sharing resources temporarily with others, and so on. Otherwise users may perceive security as a hindrance and bypass it. With delegation, the delegated user has the privileges to react to situations or access information without referring back to the delegating user. The basic idea behind the delegation is that some active entity in a system delegates authority to another active entity to carry out some functions on behalf of the former. For example, when an agent is unable to perform a task due to sickness, s/he may delegate the privileges to another agent so that the latter agent can use the privileges to complete the task on time. It is through the delegation that the agent is able to function effectively. Normally, a $delegator$ in a delegation is an agent that delegates a certain task to

another agent or a group of agents. The delegator has the permission to perform a certain action and also the ability to further delegate this right. A $delegatee$ is the one who has been delegated to execute a delegated task. A number of models dealing with various aspects of delegation have been published and been proven to be a flexible and useful access control for information sharing on a distributed collaborative environment [87, 43, 91, 71, 44, 103]. Gasser and McDermott addressed user-to-machine delegation [43]. Stein explored delegation and inheritance on the object-oriented environment [91]. Nagaratnam and Lea introduce process-to-process delegation in the distributed object environment [71]. Sandhu *et al.* addressed delegation among the role administrators in the ARBAC97 model [87]. Goh and Baldwin dealt with delegation as an attribute of roles [44]. Delegation is also applied in decentralize trust management [16, 57]. Blaze *et al.* [16] identified the trust management problem as a distinct and important component of security in network services and Li *et al.* [57] made a logic-based knowledge representation for authorization with tractable trust-management in large-scale, open, distributed systems. Delegation was used to address the trust management problem including formulating security policies and security credentials, determining whether particular sets of credentials satisfy the relevant policies, and deferring trust to third parties. Zhang *et al.* [7, 8, 117] proposed a rule-based framework for role-based delegation including RDM2000 model, which supports a user-user delegation primarily based on roles. The PBDM [115] model supports both role and permission level delegation. Wang *et al.* [100] proposed a role-based delegation model which support user-group delegation. However, all of these do not support ability-based user-user delegation and ability-based user-group delegation.

Actually, in many situations, a delegator wants to delegate a collection of permissions (named an ability) to delegatees. There may be some problems arising in the previous delegation models. For example, the permission of opening a bank account is composed of

many different individual permissions, such as, accessing identification, social secure history, credit limits and so on. When the delegator wants to delegate his ability (opening an account) to others, it does not make sense to delegate only part of the permissions to delegatees, since the entire set is needed to do the task properly. If the number is huge, it may be difficult to complete the delegation with PBDM, since each time only one permission can be assigned to a role in a permission-based delegation model. In this chapter, we propose a new delegation model, named an ability-based delegation model, which can solve this problem easily. We focus exclusively on this ability-based delegation framework, which provides great flexibility in authority management.

The rest of this chapter is organized as follows: In Section 3.2, we propose a developed assignment framework which includes group-role assignment and ability-role assignment. Our new introduced ability-based delegation model (ABDM) including delegation granting and revocation models are given in Section 3.3. We compare our work with other works in Section 3.4 and summary the chapter in Section 3.5.

3.2 ABILITY, GROUP AND AUTHORIZATION ASSIGNMENT

Role-based access control (RBAC) involves individual users being associated with roles as well as roles being associated with permissions (each permission is a pair of objects and operations). As such, a role is used to associated users and permissions. A user in this model is a human being. A role is a job function or job title within the organization associated with authority and responsibility. A permission is an approval of a particular operation to ba performed on one or more objects. When we want to open a bank account, many different individual permissions are involved. It does not make sense to assign only part of the permissions to a role, since the entire set is needed to do the task properly. The idea is that application developers package permissions into collections, named ability, which must be

assigned together as a unit to a role. Once the notion of ability is introduced, by analogy there should be a similar concept on the user side.

An *ability* is a collection of permissions that should be assigned as a single unit to a role. We denote B as the set of abilities.

A *group* is a collection of users who can accept a role assignment at the same time. Such a group can be viewed as a team. We denote G as the set of groups.

In comparison to the previous permission-role assignment, ability-role assignment is more difficult to achieve. Since if there is a huge number of permissions involved in an ability, we have to do an excessive number of jobs in order to finish the ability-role assignment, because each time we can assign just one permission to the role according to the previous permission-role assignment. So it is desired to assign all of the permissions in the ability to the role at once. The same situation happens when we want to define a group with a large number of users for role assignment. Since the function of an ability (or a group) is to collect permissions (or users) together so that administrators can treat them as a single unit, assigning abilities to roles and groups to roles are therefore very much like permission-role assignment and user-role assignment. In this way, the problems stated above can be resolved easily.

A prerequisite condition is an expression using Boolean operators '$\wedge$' and '$\vee$' on terms of the form r and $\overline{r}$ where r is a role and '$\wedge$' means 'and', '$\vee$' means 'or'. A prerequisite condition is evaluated for a user u by interpreting r to be true if $(\exists r' \geq r), (u, r') \in UA$ and $\overline{r}$ to be true if $(\exists r' \geq r), (u, r') \notin UA$, where UA is a set of user-role assignments.

Note that the notion of a prerequisite condition is identical to that, expect the boolean expression is now evaluated for membership and nonmembership of an ability (or a group) in specified roles. For a given set of roles R, let CR denotes all prerequisite conditions that can be formed using the roles in R. AR is the set of administrative roles. This leads to the

following definitions.

Definition 3.1. *Ability-role assignment and revocation are, respectively authorized by*

$can\_assigna \subseteq AR \times CR \times 2^R$

$can\_revokea \subseteq AR \times 2^R$

The meaning of $can\_assigna(x, y, Z)$ is that a member of the administrative role x can assign an ability whose current membership satisfies the prerequisite condition y to regular roles in range Z. The meaning of $can\_revokea(x, Y)$ is that a member of the administrative role x can revoke membership of an ability from any regular role $y \in Y$.

Definition 3.2. *Group-role assignment and revocation are, respectively, authorized by*

$can\_assigng \subseteq AR \times CR \times 2^R$

$can\_revokeg \subseteq AR \times 2^R$

The meaning of $can\_assigng(x, y, Z)$ is that a member of the administrative role x can assign a group whose current membership satisfies the prerequisite condition y to regular roles in range Z. The meaning of $can\_revokeg(x, Y)$ is that a member of the administrative role x can revoke membership of a group from any regular role $y \in Y$. To identify a role range within the role hierarchy, the following closed and open intervals are used.

$[x, y] = \{r \in R | x \geq r \wedge r \geq y\}$ $(x, y] = \{r \in R | x > r \wedge r \geq y\}$

$[x, y) = \{r \in R | x \geq r \wedge r > y\}$ $(x, y) = \{r \in R | x > r \wedge r > y\}$.

3.3 ABILITY-BASED DELEGATION MODEL (ABDM)

In this section we propose our flexible ability-based delegation model which supports role hierarchy. An intuitive overview of this model is described first, and then a formal definition will be presented.

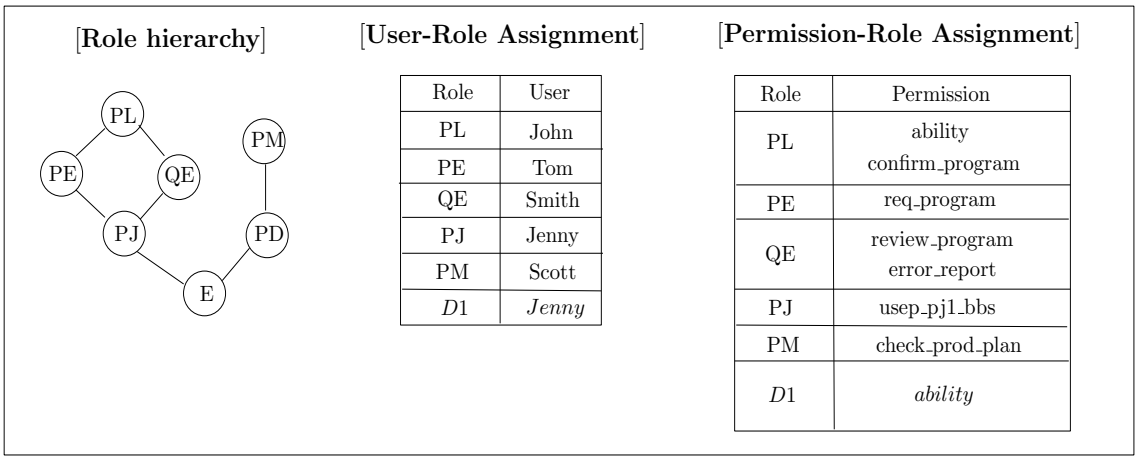


Figure 3.1: Example of ability delegation

3.3.1 ABILITY-BASED USER-USER DELEGATION

The central idea of this model is to create one or more delegation roles (DTR), and assign abilities to them. In RBAC, permissions are associated with roles, and users are assigned to appropriate roles thereby acquiring the roles' permissions. For example, in Figure 3.1 John who is in role PL acquires an ability b and a permission p. If John wants to delegate his ability b to Jenny, he can delegate according to the following three phases.

1. John creates a temporary delegation role D_1.

2. John assigns the ability b to D_1 with ability-role assignment.

3. John assigns Jenny to D_1 with user-role assignment.

Roles in DTR are distinct from regular roles (RR). DTR cannot be assigned to any other roles, because it will generate invalid ability inheritance in the role hierarchy. Therefore, roles in this model are partitioned into regular roles (RR) and delegation roles (DTR). This partition induces a parallel partition of UA and BA which are user-role assignment and ability-role assignment respectively. UA is separated into user-regular role assignment (UAR) and user-delegation role assignment (UAD). BA is similarly separated into ability-regular role assignment (BAR) and ability-delegation role assignment (BAD). Delegation

role can be placed in the regular role hierarchy when the delegated ability includes all the permissions of a delegating role, otherwise it is isolated from the hierarchy. Delegation role cannot have any senior regular role if it is placed in the role hierarchy, since delegated abilities cannot be inherited through role-role hierarchy.

We have the following components for ability-based delegation model:

Sets: U, B, R, RR, DTR are sets of users, abilities, roles, regular roles, and delegation roles respectively.

$R = RR \cup DTR$

$UAR \subseteq U \times RR$ is a user to a regular role assignment relation.

$UAD \subseteq U \times DTR$ is a user to a delegation role assignment relation.

$UA = UAR \cup UAD$

$BAR \subseteq B \times RR$ is an ability to a regular role assignment relation.

$BAD \subseteq B \times DTR$ is an ability to a delegation role assignment relation.

$BA = BAR \cup BAD$

Abilities: $R \rightarrow 2^B$ is a function mapping a role to a set of abilities.

$Abilities(r) = Abilities\_R(r) \cup Abilities\_D(r)$

where

$Abilities\_R(r) = \{b | \exists r' < r, (b, r') \in BAR\}$

$Abilities\_D(r) = \{b | \exists r' < r, (b, r') \in BAD\}$

$senior(r) : P \rightarrow 2^R$, a function mapping a role to all its senior roles in role hierarchy.

$\forall dtr \in DTR, senior(dtr) \cap RR = \emptyset$: for each delegation role there is no senior regular role.

$own(u) : U \rightarrow 2^{DTR}$ and $\nexists(u_1, u_2 \in U, dtr \in DTR), (u_1 \neq u_2) \wedge (dtr \in own(u_1) \wedge dtr \in own(u_2))$, a function mapping a user to a set of delegation roles which he/she created.

$ability\_d(r) : DTR \rightarrow 2^B$, a function mapping a delegation role to a set of abilities.

$ability^*(u)$: a function mapping a user to a set of abilities with BAD.

$ability^*(u) = \{b \in B | \exists r \in DTR, (u, r) \in UAD \wedge (b, r) \in BAD\}$

A delegation relation in ability-based user-user delegation model is a constraint on UAD and BAD.

3.3.2 ABILITY-BASED USER-GROUP DELEGATION

Now we analyze group delegation. In some cases, we may need to define whether or not a user can delegate an ability to a group and how many times, or up to the maximum delegation depth. We only analyze one-step group delegation in this paper which means the maximum delegation path is 1. Figure 3.2 shows the role hierarchy structure of RBAC in an example of a problem-oriented system POS which has two projects. In Figure 3.3 Tony who is in role Co2 acquires all the permissions and abilities of role Co2. Now Tony wants to delegate one of his abilities to $Project$ 1, which means Tony wants to delegate the ability to all people involved in $Project$ 1. According to the ability-based user-user delegation, in the third step we have to use user-role assignment. If the number of users in $Project$ 1 is small, it may be easy to finish, otherwise, it is hard to finish the work as each time just one user can be assigned to the role. It will be time-consuming if based on user-user delegation. To solve the problem, we propose an ability-based group delegation framework , in which Tony can finish the delegation according to following steps:

1. Tony creates a temporary delegation role D_1.

2. Tony assigns the ability b to D_1 with ability-role assignment.

3. Tony assigns all user of $Project$ 1 to D_1 with group-role assignment.

In fact, ability-based group delegation is achieved by ability-role assignment and group-role assignment. Previously, a delegation role cannot have any senior roles since delegated abilities cannot be inherited. Roles in this model are partitioned into regular roles RR and

Delegation Algorithm

Input: delegator u, ability b^*, delegatee.
Output: true if delegator u can delegate an ability b^* to delegatee; false otherwise.

Step 1: /*Delegator creates a temporary delegation role D_1 */
Suppose $(u, r) \in UA$ which means that delegator u is in role r,
Let $Pool_{withrole\_r} = \{p|(p, r) \in PA\} \cup \{b|(b, r) \in BA\}$,
/*$Pool_{withrole\_r}$ is the set of all permissions and abilities which are related to role r*/
If $\{b^*\} = Pool_{withrole\_r}$ /*role r only has an ability b^**/
Delegator can let role r to be the temporary delegation role D_1.
go to Step 3
If $\{b^*\} \subset Pool_{withrole\_r}$
/*role r not only has the ability b^* but also other permissions*/
Delegator creates a temporary delegation role D_1.
go to Step 2
else
return false and stop.
Step 2: /*whether the delegator can assign the abilityb^* to delegation role D_1 or not*/
Let $S = \pi_{RoleRange}(\sigma_{delegator}(can\_assigna))$
/*S is the role range where the ability b^* can be assigned to*/
If $D_1 \in S$, /* the delegation role is in the role range*/
go to Step 3.
/* the ability b^* can be assigned to D_1 by delegator*/
else
return false and stop.
Step 3: /*whether the delegator can assign the delegatee to the delegation role D_1*/
Suppose the delegatee is a user u',
Let $S = \pi_{RoleRange}(\sigma_{delegator}(can\_assign))$,
/* S is the role range where the user can be assigned to*/
If $D_1 \in S$, /* the delegation role is in the role range*/
the user u' can be assigned to the delegation role D_1.
Suppose the delegatee is a group g,
Let $S = \pi_{RoleRange}(\sigma_{delegator}(can\_assigng))$,
/* S is the role range where the group can be assigned to*/
If $D_1 \in S$, /* the delegation role is in the role range*/
the group g can be assigned to the delegation role D_1.
return true;
else
return false.

RoleName	Prereq.Condition	M
HO2	[*AP, Ho1*]	1
Co1	*CS*	2

Table 3.1: Example of $can\_delegatea$

RoleName	RoleRange
Ho1	[*Co1, CS*]
Re1	[*AP, AP*]

Table 3.2: Example of $del\_revokea$

delegation roles (DTR). This partition induces a parallel partition of GA and BA which are group-role assignment and ability-role assignment respectively. GA is separated into group-regular role assignment (GAR) and group-delegation role assignment (GAD). BA is similarly separated into ability-regular role assignment (BAR) and ability-delegation role assignment (BAD). Hence we have the following elements and functions in group delegation:

Sets: U, G, B, R, RR, DTR are sets of users, groups, abilities, roles, regular roles, and delegation roles respectively.

$R = RR \cup DTR$

$GAR \subseteq G \times RR$ is a group to a regular role assignment relation.

$GAD \subseteq G \times DTR$ is a group to a delegation role assignment relation.

$GA = GAR \cup GAD$.

$BAR \subseteq B \times RR$ is an ability to a regular role assignment relation.

$BAD \subseteq B \times DTR$ is an ability to a delegation role assignment relation.

$BA = BAR \cup BAD$

Abilities: $R \rightarrow 2^B$ is a function mapping a role to a set of abilities.

$Abilities(r) = Abilities\_R(r) \cup Abilities\_D(r)$

where

$Abilities\_R(r) = \{b | \exists r' < r, (b, r') \in BAR\}$

$Abilities\_D(r) = \{b|\exists r' < r, (b, r') \in BAD\}$

$senior(r) : P \rightarrow 2^R$, a function mapping a role to all its senior roles in role hierarchy.

$\forall dtr \in DTR, senior(dtr) \cap RR = \emptyset$: for each delegation role there is no senior regular role.

$own(u) : U \rightarrow 2^{DTR}$ and $\nexists(u_1, u_2 \in U, dtr \in DTR), (u_1 \neq u_2) \wedge (dtr \in own(u_1) \wedge dtr \in own(u_2))$, a function mapping a user to a set of delegation roles which he/she created.

$ability\_d(r) : DTR \rightarrow 2^B$, a function mapping a delegation role to a set of abilities.

$ability^*(g)$: a function mapping a group to a set of abilities with BAD.

$ability^*(g) = \{b \in B|\exists r \in DTR, (g, r) \in GAD \wedge (b, r) \in BAD\}$

A delegation relation in an ability-based user-group delegation model is a constraint on GAD and BAD.

The delegation algorithm described above provides a way for the delegator to delegate an ability to the desired delegatee.

3.3.3 ABILITY-BASED DELEGATION AUTHORIZATION

In this section, we develop the delegating and revocation models. The goal of the delegation authorization is to impose restrictions on which role can be delegated to whom. Here, we partially adopt the notation of prerequisite condition from [103] to introduce delegation authorization in the delegation framework.

Definition 3.3. *$can\_delegatea$ is a relation of $RR \times CR \times M$ where RR, CR, M are sets of regular roles, prerequisite conditions, and maximum delegation depth, respectively.*

The meaning of $can\_delegatea(r, cr, m)$ means a delegator having regular role r who can delegate an ability to any user or group whose current entitlements in role satisfy the prerequisite condition cr without exceeding the maximum delegation depth m. Table 3.1

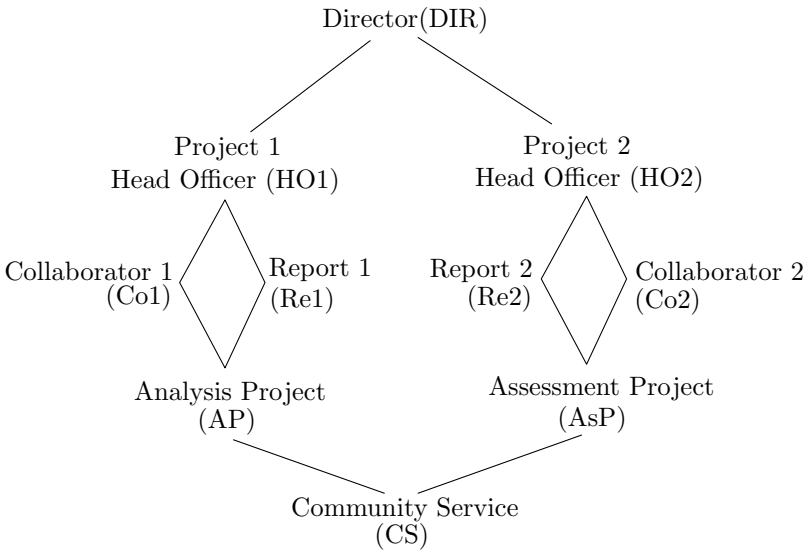


Figure 3.2: Role hierarchy in POS

[User-Role Assignment]

role	user
Co2	Tony
D1	*Group* {all users in *project 1*}

[Permission-Role Assignment]

role	permission
Co2	*ability* error_report
D1	*ability*

Figure 3.3: Example of Group Delegation

shows the $can\_delegatea$ relations with the prerequisite conditions in the POS example. The meaning of $can\_delegatea(Ho2, [AP, Ho1], 1)$ is that a user of role $Ho2$ can delegate his/her ability to a group in which users are members of either role AP, or $Co1$, or $Re1$, or $Ho1$. In addition, the delegated ability cannot be re-delegated to other users or groups where the maximum depth of delegation is 1. The second tuple authorizes that a user of role $Co1$ can assign an ability to another user who has CS role and the delegated ability can be re-delegated to other users or groups with the maximum depth of delegation of 2.

Definition 3.4. *An ability-based delegation revocation is a relation* $del\_revokea \subseteq RR \times$

2^R*, where* RR *is the set of regular roles.*

The meaning of $(x, Y) \subseteq del\_revokea$ is that a delegator who has regular role x can revoke the relationship of a user or group from any role $y \in Y$, where Y defines the range of revocation. Table 3.2 gives the $del\_revokea$ relation in Figure 3.2. The first tuple shows that the delegator who has the role $Ho1$ can revoke a delegation relationship of a group from any role in $[Co1, CS]$. The second tuple shows that the delegator who has the role $Re1$ can revoke a delegation relationship of a user from role in AP.

Actually, the revocation process can be finished through any of the following cases:

1. Revoke the user-delegation role assignment or group-delegation role assignment.
2. Revoke the ability-delegation role assignment.
3. Revoke the delegation role.

3.4 RELATED WORK AND COMPARISONS

The close work to this paper is role-based delegation [8], role-based group delegation [100] and permission-based delegation [115].

Barka and Sandhu [8] proposed a simple model for role-based delegation called RBDM0. They developed a framework for identifying interesting cases that can be used for building role-based delegation models. This is accomplished by identifying the characteristics related to delegation, using these characteristics to generate possible delegation cases. Their work is different from ours in three aspects. First, the unit of delegation in RBDM0 is 'role', which means the delegator can just delegate roles to delegatee. Whereas, our work supports a piece of role delegation not only of some permissions but also a unit of permissions, called ability. Second, [8] focuses on a simple user-user delegation model supporting only flat roles and single step delegation. By contrast, our work develops users to groups and proposes

a user-group delegation model based on ability. Third, some important features such as role hierarchies, constraints and revocations are not supported. But, our work has analyzed delegation authorization and revocation models with constraints involving role hierarchies.

In [100], the authors proposed a role-based group delegation model in which a delegating user can delegate a role to a group at one time. Our work totally differs from that. The authors in [100] did not discuss partial role delegation. In some situations, a delegator may want to delegate permissions rather than roles to a delegatee. Moreover, our model not only supports permission-based delegation but also extends it to ability-based delegation.

Permission-based delegation model(PBDM) is first proposed in [115]. PBDM supports user-user delegation based on permissions and roles. In our work, we extend their work and propose ability-based delegation. If we restrict an ability to a permission, our model will reduce to PBDM. Moreover, [115] does not mention one-to-many delegation, whereas, we discuss the user-group delegation specifically. Finally, some important features such as authorization, constraints and revocations are discussed shortly in PBDM, whereas we analyze delegating framework including delegation authorization and revocation with constraints.

3.5 SUMMARY

The main contribution of this chapter is to introduce a new and flexible ability-based delegation framework within RBAC. Moreover, we extend the framework to include both delegation granting and revocation. The work presented in this chapter has significantly extended previous work, which provides a flexible and useful management of delegation authority in a role-based access control environment.

CHAPTER 4

MULTI-LEVEL DELEGATIONS WITH TRUST MANAGEMENT

In this chapter, we decompose delegation tasks into three different levels according to the sensitivity of each delegation task. Each level has different requirement of reliability of cooperation partners. We also propose a new effective trust evaluation technique which considers both trust values and trust trend. The trust value provides an indication for the final trust level while, the trust trend value is used to predict the future trend of trust. We build a projection between the reliability of the delegatee and the sensitivity of delegated tasks, which leads to a secure multi-level delegation model.

The information in this chapter is based on a published paper [66].

4.1 INTRODUCTION

Role-based delegation based on role-based access control (RBAC) has been proven to be a flexible and useful access control for information sharing in distributed collaborative environments [8, 115, 70]. In contrast to normal access right administration operations which are performed centrally, delegation operations are usually performed in a distributed manner. Security of delegation becomes one big issue that has received attention during the past few years in distributed systems. In this section, we are interested in the delegation of tasks (task-delegation) as compared with the delegation of rights only (right-delegation) described in [1]. Both task-delegation and rights-delegation involve the release of rights from one principal to another. However, in the case of task-delegation, we consider the situation in

which an entity issues an imperative command to another entity to perform the delegated task within the broad area of security.

Our line of reasoning is motivated by the real-world situations in which one entity delegates some rights to a second entity with the explicit command to complete a given task validly and securely. Loosely defined, a task consists of a number of computational operations to be performed based on some data which may be sensitive, and if insecure may be misused or disclosed to public. Here, we organize delegated tasks into three different levels according to their sensitivity as shown in Table 4.1. For simplicity of discussion, in this paper we consider three-level partitions of delegated tasks, which are Low, $Medium$, and $High$. The classification standard is flexible, which can be determined by a delegator with his or her subjective preference. A Low task level indicates the delegation task does not include sensitive information or resources that can cause a breach. The information in a Low level of the delegation task is public information that can be delegated to anyone for information sharing. The tasks in the $Medium$ level contain information that is partially public and partially sensitive. When referring to the delegation, there should be a higher requirement on the reliability of the delegatee, since the more reliable the delegatee is, the less chance the sensitive information would be misused, and the more likely the delegation task could be accomplished successfully. The $High$ task level indicates the delegation task is very important and contains highly sensitive information and requires that the delegatee should be totally trustworthy.

Essentially, a delegation operation could temporarily change the access control state so as to allow an agent to use another agent's access privileges. Due to its effect on the access control state, delegation may lead to violation of security policies. More precisely, information breaching may happen even during the delegation phase. This refers to any information, the loss, misuse, or unauthorized access to or modification of which could adversely affect

the privacy to which individuals are entitled. Thus, risk during the delegation must not be overlooked, and more sophisticated methods are needed to create a secure delegation system. More specifically, delegation policies may depend on private aspects concerning both the delegatee's reliability and the sensitivity of the delegated tasks.

The remainder of this chapter is structured as follows. In Section 4.2, we describe the motivation of this paper. In Section 4.3, the new trust evaluation approach is proposed by combining trust values and trust trend together to predict a delegatee's trustworthiness. In Section 4.4, we propose a multi-level delegation model with trust management and discuss several different delegation types. We show our experimental results in Section 4.5 and provide a brief survey of related work in Section 4.6. Finally, we summary this chapter in Section 4.7.

Task level	Information	Properties
Low	Public	The information is not sensitive and can be delegated to anyone.
Medium	Not public partially sensitive	The information is partially sensitive and should be delegated to reliable delegatees.
High	Not public totally sensitive	The information is totally sensitive and should be delegated to someone with higher reliability.

Table 4.1: The classification of delegation tasks

4.2 MOTIVATION

In an open environment, the entities are customarily alien to each other. When entering into a delegation, the delegator is entering into an uncertain interaction in which there is a risk of failure due to the delegation decisions. In other words, a given delegatee may not be reliable for the delegated task, especially, when sensitive information is included in the delegation tasks, the delegator's privacy may be breached because of the unreliability of the delegatee. For example, if the task being delegated is a goal comprising of multiple tasks and requir-

ing access to multiple resources and sensitive information, the delegation in this case should be very cautious, since the failure of the delegation has considerable influence on privacy disclosure. Therefore, when delegating a task, the choice of the cooperative partner plays an important role in determining whether the task would be fulfilled successfully or not. In order to operate effectively, delegators need some mechanisms for finding reliable partners, and this requirement could be satisfied with the help of trust. Trust is well recognized as a means of assessing the risk of cooperating with others [31, 55, 68, 111]. There are two main categories of trust: experience-based and recommendation-based [81, 113]. In the former category, agents assess the trust solely based on their own experience; in the latter, trust is evaluated based on information provided by others (typically in addition to individual experience). Within this trust evaluation mechanism, a final trust value is computed to reflect the general trust status of every service provider. However, such a single trust value cannot reflect the real trust status very well. For example, assume the trust values are in the range of [0,1]. A person with a higher trust value 0.9 may behave worse in future than the one with the trust value 0.6. This simple example demonstrates that the single-value trust evaluation approach can not reflect future changes in trust. Trust trend evaluation becomes important in order to indicate whether the trust will become better or worse in the forthcoming cooperation. Therefore, new effective trust evaluation approaches are required to provide more precise trust information that could indicate to what extent and during which period a delegatee is reliable and trustworthy.

Even though delegation is well recognized as a very useful component of access control systems [8, 30, 115], to our best knowledge, no current work has performed an in-depth study on how to manage a delegation in a secure manner. Typically, we are facing the following challenges in developing a secure multi-level delegation model by taking trust into account:

Challenge One: Since the sharing of sensitive information must be restricted to trust-

worthy parties, how to develop effective trust evaluation approaches to provide more precise trust information? The developed method should not only predict the trust value but also capture its future trend.

Challenge Two: Faced with the fact that delegation tasks in different levels require different reliability of delegatees, how to build the projection between the reliability of the delegatees and the sensitivity of the delegated tasks, and further construct a secure multi-level delegation model?

4.3 TRUST EVALUATION

The notion of trust is well recognized as a means of assessing the risk of cooperating with others [31, 111, 105]. In a delegation, it is important to tell delegators to what extent a delegatee is trustworthy for the delegated task. Corresponding to the different levels of delegated tasks, in this section we organize the trust into three trust levels, in which delegators could evaluate the trustworthiness of delegatees.

Trust represents an agent's estimate of how likely another is to fulfil its commitments. Trust influences the delegators attitudes and actions, but can also have effects on the delegatee and other elements in the environment. As discussed before, a trust value can be calculated to provide more precise indication of the trust history to a delegator. However, it is not enough to indicate the real trust status of a delegatee very well, i.e., the single-value approach cannot reflect changes of the trust trend. In this paper, we adopt two interpretations of trust. One is to view trust as the perceived reliability history of somebody, called "reliability trust", while the other is to view trust as a trend of trust changes in a given period, called "future trust".

Definition 4.1 (Reliability trust). *Reliability trust is the trust status of individuals depen-*

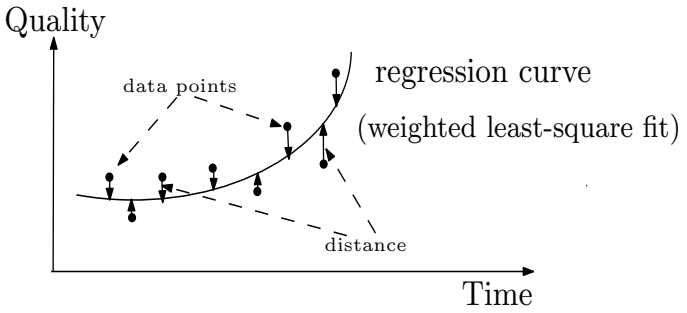


Figure 4.1: Weighted least-squares exponential regression

dent on his/her history behavior.

As the name suggests, reliability trust can be interpreted as the subjective probability of someone performing a given action on which its success lies. In our previous work [60], we evaluated reliability trust in three steps: (1) Calculate the trust value based on histories; (2) Calculate the trust value from recommendations; (3) Combine the observed trust values from histories and recommendations. With this approach, we can obtain a delegatee's reliability trust value. However, trust can be more complex. Future trust aims to capture the changes of trust trend in the forthcoming future. Namely, given a set of delegatees with the same trust value, the one which is becoming better is more desirable to delegators and more reliable to fulfill the delegated work well.

Definition 4.2 (Future trust). *Future trust is a general trend of changes which could be useful to predict the future trust level of service quality.*

In order to evaluate future trust, we refer to the idea of exponential regression [99]. In this section, we introduce a weighted exponential regression method to evaluate the trust trend (shown as Figure 4.1). This method is used to obtain the best exponential fit from a set of given data points. This best exponential fit is characterized by the sum of weighted squared residuals with its least value, where a residual is the difference between a data point and the

regression curve. Once obtaining the exponential regression, the gradient at each data point can be taken as our future trust value. Now we introduce the trust trend evaluation method.

Let $(t_1, q_1), (t_2, q_2), \ldots, (t_n, q_n)$ denote the given data points in a certain period, where $q_i (q_i \in [0, 1])$ is the service quality value at time $t_i (t_i < t_{i+1}, 1 \leq i \leq n)$. Then the exponential regression can be represented as

$$q = a_0 e^{a_1 t} + a_2 \tag{4.1}$$

where a_0, a_1 and a_2 are constants to be determined, specially, the product of a_0 and a_1 indicates the trust trend value. As the distance from point (t_i, q_i) to the regression curve is

$$d_i = |q_i - (a_0 e^{a_1 t_i} + a_2)| \tag{4.2}$$

Based on the method of weighted least squares, we let $w(i)$ be the weight function for the service quality q_i at the i^{th} service $(i = 1 \cdots n)$. The choice of $w(i)$ could be flexible. Any monotonic increasing function could be a candidate of $w(i)$. For simplicity, in this paper, we adopt $w(i) = i^{\beta}$, $(1 \leq i \leq n, \beta \geq 1)$ as our weight function. Thus, the sum of squares of the distance can be calculated as follows:

$$S = \sum_{i=1}^{n} w(i)^2 d_i^2 = \sum_{i=1}^{n} w(i)^2 (q_i - (a_0 e^{a_1 t_i} + a_2))^2 \tag{4.3}$$

Now our task is to minimize the sum of the distance S with respect to the parameters a_0, a_1 and a_2, with the method of undetermined coefficients.

Since function S is continuous and differentiable, based on the Lagrange Multiplier method [97], the minimization point of S makes the first derivative of function S be zero.

Thus, we differentiate S with respect to a_0, a_1 and a_2, and set the results to zero, which gives

$$\frac{\partial S}{\partial a_1} = -2\sum_{i=1}^{n} w(i)^2(q_i - (a_0 e^{a_1 t_i} + a_2))(a_0 t_i e^{a_1 t_i}) = 0 \tag{4.4}$$

$$\frac{\partial S}{\partial a_2} = -2\sum_{i=1}^{n} w(i)^2(q_i - (a_0 e^{a_1 t_i} + a_2)) = 0 \tag{4.5}$$

and

$$\frac{\partial S}{\partial a_0} = -2\sum_{i=1}^{n} w(i)^2(q_i - (a_0 e^{a_1 t_i} + a_2))e^{a_1 t_i} = 0 \tag{4.6}$$

Equations (4.4), (4.5) and (4.6) can be solved for the unknown a_0, a_1 and a_2. Thus, based on the method of weighted least squares exponential regression, we can obtain the trust trend value $a_0 a_1 (a_0, a_1 \in R)$. The trust trend value shows a general trend of changes of trust in the near future, which is important when we choose a delegatee with serious caution. If $a_0 a_1 > 0$, it indicates that the future trust is up-going, whereas, $a_0 a_1 < 0$ indicates that the future trust is dropping; and $a_0 a_1 = 0$ indicates the future trust remains unchanged.

Both reliability trust and future trust reflect different trust status about the individuals on whom the delegator depends for the delegation task. Reliability trust is most naturally measured as a degree of reliability, which is expressed as a continuous function mapped into [0,1], whereas future trust indicates the trend of trust changes, which ranges from $-\infty$ to $+\infty$. To work efficiently, we combine reliability trust and future trust into different trust levels to illustrate the trustworthiness of a delegatee. To be consistent with delegated task levels, three trust levels are organized through the following projection:

Definition 4.3 (Trust level). *Let T be the set of reliability trust values and TT be the set of future trust values. The F function projects reliability trust and future trust into three*

different trust levels.

$$F : T \times TT \rightarrow \{L, M, H\}$$

where L, M, H refers to Low, $Medium$ and $High$ trust levels.

$High$ trust level denotes the person at this level is highly trusted, which means not only his final trust value is high but also the trust trend is up-going. Low level denotes the person is less trusted, where his final trust value is low, also the trust trend is dropping. $Medium$ level is the intermediate state. So the trust level assignments can be further explained as follows:

$\forall t \in T, a_0a_1 \in TT$

- $F(t, a_0a_1) = L$, if $t \in (0, 0.5)$ and $a_0a_1 \in (-\infty, 0)$
- $F(t, a_0a_1) = M$, if $t \in [0.5, 1)$ and $a_0a_1 \in (-\infty, 0]$; or $t \in (0, 0.5]$ and $a_0a_1 \in [0, +\infty)$
- $F(t, a_0a_1) = H$, if $t \in (0.5, 1)$ and $a_0a_1 \in (0, +\infty)$

Until now, each delegatee is companied with a trust level, which could indicate to what extent the delegatee is reliable. So far, the problem left is to build the delegation model based on the evaluation of trust. Our idea is that the delegatee who is trusted to a greater degree would have a higher probability of completing the delegated task than a delegatee with a lower trust level. The formalized delegation model is described in the next section.

4.4 THE MULTI-LEVEL DELEGATION MODEL

Delegation has received significant attention from the research community in recent years. A number of delegation models have been proposed [30, 53, 98, 59] and most of them are for Role-Based Access Control (RBAC). A few research works related to introducing subjective

trust into a delegation model have been reported [46, 110]. In this section, we build a multi-level delegation model with trust management.

4.4.1 THE DELEGATION MODEL

In a multi-agent system, delegation is the primary mechanism of inter-agent collaboration and cooperation [47, 53, 59, 77]. Delegation is a mechanism that allows an agent A to act on another agent B's behalf by making B's access rights available to A. Suppose a task is delegated from one to another, the latter actually gets the access right to work on this task. It needs to be organized as an important mechanism to provide resiliency and flexibility in access control systems.

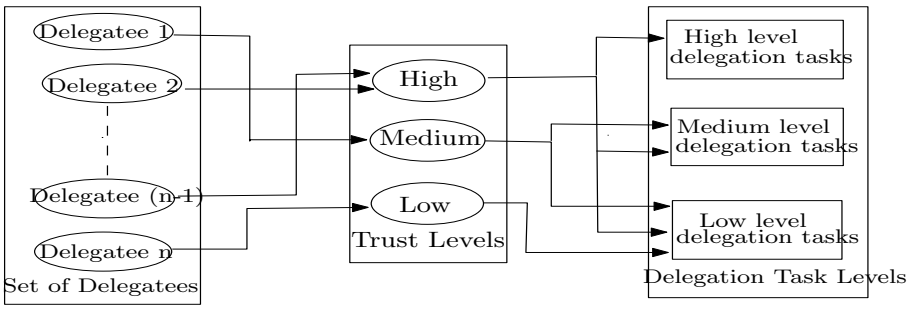


Figure 4.2: Distribution of delegations based on trust levels

Since delegation tasks are divided into three different levels, it is important to address how to distribute these tasks to delegatees based on their trust levels. The idea is that a delegatee in the high trust level can be assigned with the delegation task of all levels, which are Low, $Medium$, and $High$. The delegatee in the medium trust level can be assigned with Low and $Medium$ level tasks, while the delegatee in the low trust level can only be assigned with Low level tasks. In this case, all delegated tasks are assigned in a hierarchal style, since the delegatee in a higher trust level is more trustworthy and is more likely to finish a higher level delegation task than the one in a lower trust level. The distribution of delegation is

shown in Figure 4.2. In order to describe the delegation in a precise manner, we focus on a specific model about how delegatees gain access rights.

Definition 4.4. *Let D_r, D_e, D_t be the set of delegators, delegatees, and delegated tasks respectively. $Level = \{L, M, H\}$ is the set of trust levels (or delegated task levels). A delegation relationship is defined as $DR \subseteq D_r \times D_eL \times D_tL \times \{g, t\}$, where $D_eL \subseteq D_e \times Level$ is the membership between delegatees and trust levels, $D_tL \subseteq D_t \times Level$ is the membership between delegated tasks and task levels, and g, t refers to a $grant$ or $transfer$ operation.*

The delegatee-trust level membership D_eL denotes that each delegatee is assigned with different trust levels and D_tL denotes that each delegated task is assigned with different task levels. For example, the delegation relationship $(d_r, (d_e, L), (d_t, M), g) \in DR$ indicates that delegator d_r has delegated the L level task d_t to delegatee d_e in the M trust level via a $grant$ operation, while $(d_r, (d_e, L), (d_t, M), t)$ indicates that delegator d_r has delegated L level task d_t to delegatee d_e in M trust level via a $transfer$ operation. The difference between $grant$ and $transfer$ is shown as follows. A delegation operation is essentially an access control state transition operation, which takes one of the following three forms:

- $grant(d_r, (d_e, l), (d_t, l))$: delegator d_r $grants$ the access of l level delegation task to delegatee d_e who is in l trust level. After the delegation operation, d_e gains the access right to d_t and d_r still keeps d_t, where $l \in \{L, M, H\}$.

- $trans(d_r, (d_e, l), (d_t, l))$: delegator d_r $transfers$ the access of l level delegation task to delegatee d_e who is in l trust level. After the delegation operation, d_e gains the access right to d_t and d_r temporarily loses d_t, where $l \in \{L, M, H\}$.

- $revoke(d_r, (d_e, l), (d_t, l))$: delegator d_r $revokes$ the delegated task d_t from delegatee d_e.

Note that a delegator can grant or transfer different level tasks to delegatees, and only the corresponding delegator can revoke the delegated task from the delegatee. For example,

$grant(Alice, (Bob, H)$, (***read all emails***, $M), g)$ means Alice delegated the $Medium$ level task "read all emails" to Bob with $High$ trust level via a $grant$ operation, while after the delegation Bob gains the access right to all emails and $Alice$ still keeps the access right on all emails. However, $transfer(Alice, (Bob, H)$, (***read all emails***, $M), t)$ means $Alice$ delegated the $Medium$ level task "read all emails" to Bob with $High$ trust level via a $transfer$ operation, and after the delegation $Alice$ temporarily loses the access right to all emails. Definitely, only $Alice$ could revoke the delegated task "read all emails" from Bob.

Since delegation is performed in a distributed manner, in the sense that everyone may perform delegation operations, it is undesirable to allow a delegator to delegate the tasks in a completely unrestricted way. Delegation operations are thus subject to the control of authorization rules, which takes one of the following three forms:

- $can\_grant(cond, (d_t, l))$: a delegator who satisfies condition $cond$ can grant the l level task d_t to other delegatees, where $l \in \{L, M, H\}$, $cond$ is an expression formed through using the binary operators $\vee$ and $\wedge$, the unary operator $\neg$, and parentheses.

- $ca\_transfer(cond, (d_t, l))$: a delegator who satisfies condition $cond$ can transfer the l level task d_t to other delegatees, where $l \in \{L, M, H\}$.

- $can\_receive(cond, (d_t, l))$: a delegatee who satisfies condition $cond$ can receive the l level task d_t from other delegators, where $l \in \{L, M, H\}$.

For example, the rule $can\_receive$(Clerk $\wedge$ M, ("read the documents", M)) states that anyone who is at the $Medium$ trust level and a member of Clerk can receive the $Medium$ level task "read the document".

4.4.2 TYPES OF DELEGATIONS

Delegation models can be complicated. To create a delegation model, one needs to decide on a number of features, such as whether the delegation is dated and valid only for a certain period of time, whether delegatees can further delegate the tasks to others and so on. Retention period refers to the time during which the delegation is valid. We denote TI as the set of time intervals. Different types of delegations contained in our delegation model are discussed as follows.

- ***Time Bound Delegation*** $TBD \subseteq TI \times D_r \times D_eL \times D_tL$: It is a delegation that is valid only for a certain time period, where T is the set of time intervals.

  For example, delegation *([12/06/2008, 10/08/2008], Alice, (Bob, H), (read all emails, M))* denotes that this delegation is only valid between 12/06/2008 and 10/08/2008 and only during this period, Bob has the access right to all emails.

- ***Group Delegation*** $GD \subseteq D_r \times D_eL \times D_tL$: It can be used to delegate access rights to a group of delegatees who satisfy certain conditions.

  For example, delegation*(Alice, (Employee, M), (read all emails, M))*denotes that Alice delegates the $Medium$ level task "read all emails" to a group of employees who are in $Medium$ trust level.

- ***Action Restricted Delegation*** $ARD \subseteq D_r \times D_eL \times D_tL \times CD$: This forces the delegatee to satisfy certain conditions before the delegated task can be carried out, where CD is the set of conditions.

  For example, delegation*(Alice, (Employee, M), (read all emails, M), (age(24), name(Bob)))* states that only employees who are in $Medium$ trust level, aged 24 and named Bob can gain the access right to "read all emails".

- ***Re-delegable Delegation*** $RD \subseteq D_r \times D_eL \times D_tL \times \{True, False\}$: In this delegation, $True$ means the delegated task could be re-delegated to others, while $False$ means not.

  For example, delegation*(Alice, (Employee, M), (read all emails, M), true)* denotes that the delegatee is allowed to further delegate the task.

Delegation policy: Delegation policies describe rules for the delegation of rights. A rule for delegation would be checking that an agent has the ability to delegate before allowing the delegation to be approved. A policy can be viewed as a set of rules for a particular domain that defines what permissions a user has and what permissions she/he can obtain. A policy also contains basic or axiomatic rights that all individuals possess.

4.5 EXPERIMENTAL EVALUATIONS

The main goals of the experiments are two-fold. First, we study the precision of our trust model in predicting the trend of the trust. Second, we investigate the effectiveness of our proposed multi-level trust-based delegation model in terms of *disclosure rate*.

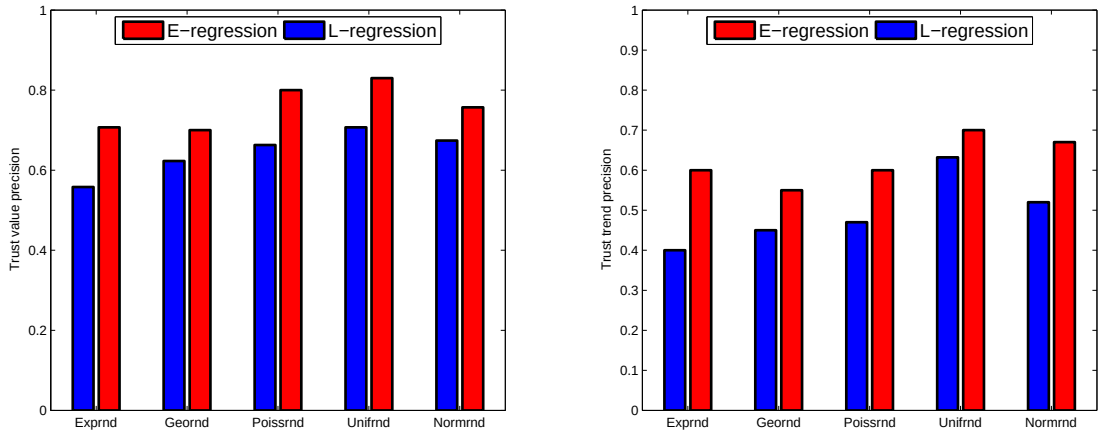
No. of data set	Probability distribution function
1	exponential distribution (Exprnd)
2	geometric distribution (Geornd)
3	Poisson distribution (Poissrnd)
4	Uniform distribution (Unifrnd)
5	Normal distribution (Normrnd)

Table 4.2: Distributions of the data sets

Trust value and its trend evaluation: In this set of experiments, we compared the precision of both the trust value and trust trend prediction with the existing method proposed in [58]. We denote *E-regression* as the exponential regression model proposed in this paper and *L-regression* as the regression model of [58]. In order to evaluate the precision of the two approaches, we generate five data sets with five different probability distribution functions

as our test data, and each data set contains 5000 records, and each record is in the form of (x, y), where $1 \leq x \leq 5000$ and $0 \leq y \leq 1$. Table 4.2 shows the probability distributions of each data set. Different metrics are adopted in evaluating the precision of the trust value and trust trend. For evaluating the precision of trust value, each data set is first divided into training and testing sets, and both regression models are trained by the training sets and tested by testing sets. If the predicted trust value is t_{pre} and the actual trust value is t_{act}, then the precision is calculated as $1 - \frac{|t_{pre} - t_{act}|}{t_{act}}$. The higher the value, the more precise the predicted trust value. To evaluate the precision of the trust trend, we use the metric named *vector angle*, which computes the angle between two vectors(trends). The vector angle is defined to be the angle ϕ between 0 and 180 degrees that satisfies the relationship: $\cos\phi = \frac{t_1 \cdot t_2}{|t_1||t_2|}$, where $|\cdot|$ refers to the vector length and the numerator denotes the inner product of the trends t_1 and t_2. The closer the cosine value is to 1, the more similar two trends are. To reduce the randomness, we run the evaluation for 1000 times for each data set to obtain the average.

The evaluation results are shown in Figure 4.3. Figure 4.3(a) displays the precision of the trust value of both regression models under five different distributed data sets. We can easily see that the average precision of our proposed exponential regression model is around 70%, which is superior to the linear regression model over all five different distributed data sets. Figure 4.3(b) reports the precision of the trust trend for both regression models. From the graph, the exponential regression model results in a more accurate trust trend compared with the linear regression model over all the five different distributed data sets. The precision of the trust trend for the linear regression model sometimes is quite low, for example, only 40% for the exponential distributed data set. This is because sometimes the linear regression model predicts the opposite trust trend, which makes the cosine value negative and hence drags down the average precision. Overall, the exponential regress model proposed in this paper has more accurate precision in predicting both the trust value and trust trend than the

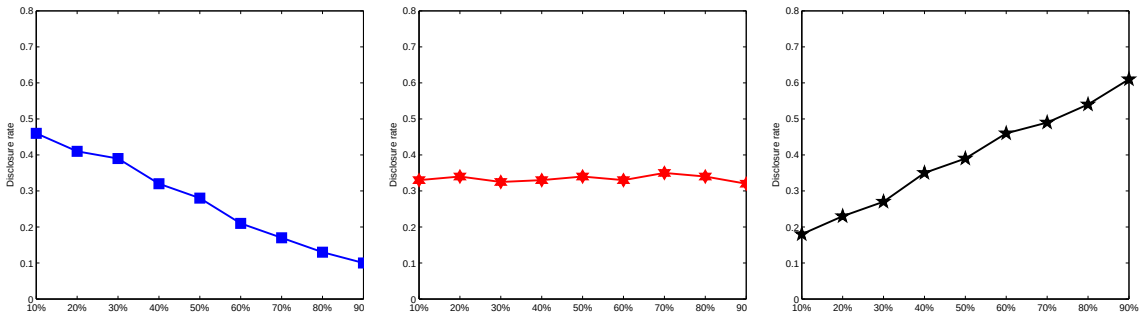


(a) (b)

Figure 4.3: (a) The precision of the trust value; (b) The precision of the trust trend.

linear regression model.

Effectiveness: Having verified the precision of our technique, we proceed to test its effectiveness. In this set of experiments, we use the *disclosure rate* to measure the effectiveness of our proposed multi-level delegation model. We are going to use H, M and L to denote the High, Medium and Low level in the classification of delegation tasks or the trust level of the delegatees, separately. Recall our trust-based delegation model, if a data requester is in High trust level, then s/he can be assigned with H, M or L level tasks; if the data requester is in Medium trust level, then s/he can be assigned with M or L level tasks; Otherwise, the data requester can only be assigned with L level tasks. Suppose there are n data requesters, among which there are n_H data requesters with High level trust, n_M requesters with Medium level of trust, and n_L with Low level of trust, where $n_H + n_M + n_L = n$. In this case, the requesters could totally access $3n_H + 2n_M + n_L$ delegation tasks, which indicates the number of secure delegations. Consider the situation where there is no specification of trust levels, the data requester, whatever the trust value and trend s/he holds, could receive three possible task assignments. Then it would be $3(n_H + n_M + n_L)$ delegations, and among those, there



(a) (b) (c)

Figure 4.4: Disclosure rate comparison when varying (a) the number of H levels; (b) the number of M levels; (c) the number of L levels.

will be $3(n_H + n_M + n_L) - (3n_H + 2n_M + n_L)$ insecure delegations. Thus, we define the *disclosure rate* as $1 - \frac{3n_H+2n_M+n_L}{3(n_H+n_M+n_L)}$. The lower the rate is, the more secure the delegation is. We randomly generate n data requesters, and evaluate how the number of data requesters in H, M or L levels affect the disclosure rate. In order to reduce the randomness, we run the test for 500 times for each data and use the average to mark the graph.

The results are shown in Figure 4.4. Figure 4.4(a) displays the disclosure rate by varying the portion of H from 10% to 90%. From the graph, we can see that the disclosure rate is decreasing as the amount of H increases. This is expected, since the more the H level requesters, the less insecure delegations there are, and the lower the disclosure rate is. Figure 4.4(b) describes the disclosure rate by varying M from 10% to 90%. The graph shows that the disclosure rate almost remains unchanged with the increased portion of M. Figure 4.4(c) reports the effect of L on the disclosure rate. When varying the portion of L from 10% to 90%, the disclosure rate is ascending. It indicates that the more L level requesters are assigned to delegation tasks, the higher the chances for sensitive information to be disclosed. However, our proposed delegation model could better avoid sensitive information disclosure by specifying requesters' trust levels. Therefore, in this case, our proposed multi-level delegation model is superior to the traditional delegation model.

4.6 RELATED WORK

Delegation has received considerable attention from the research community. In [8], Barka and Sandhu proposed a framework for role-based delegation models (RBDM), which identifies a number of characteristics related to delegation. Example characteristics are monotonicity, totality, and levels of delegation.

There exists a wealth of delegation models in literature [114, 115, 30]. Zhang et al. [114] presented a role-based delegation model called RDM2000. Their model supports the specification of delegation authorization rules to impose restrictions on which roles can be delegated to whom. Zhang et al. [115] proposed a role-based delegation model called PBDM, which supports both role and permission level delegation. Their model controls delegation operations through the notion of delegatable roles such that only permissions assigned to these roles can be delegated to others. In [30], Crampton and Khambhammettu proposed a delegation model that supports both grant and transfer. Atluri and Warner [6] studied how to support delegation in workflow systems. They extended the notion of delegation to allow conditional delegation, where conditions can be determined on time, workload and task attributes. One may specify rules to determine under what condition a delegation operation should be performed.

All of the above work focuses on the modeling and management of delegation, while our paper focuses on developing a secure delegation model in access control systems. More importantly, none of the above work discusses the trust relationship between delegators and delegatees, but our delegation model is founded on trust. We also investigate the effectiveness of our proposed multi-level delegation model and the experimental results confirm the advantages of our model in privacy protection.

Trust evaluation is a recent approach for access control systems that enables resource requesters and providers in open systems to establish trust. Bonatti and Samarati [21] proposed

a framework based on a policy language and an interaction model for regulating access to network services. Their trust establishment framework uses logical rules for accessing services and avoiding the unnecessary disclosure of sensitive information. Winsborough and Li [109] introduced the Trust Target Graph (TTG) protocol for conducting trust negotiation. A particular emphasis of their work was protection against leaking sensitive information during a trust negotiation. PeerTrust [78] is a trust management system that uses a simple and expressive policy language based on distributed logic programs. PeerTrust agents perform automated trust negotiation to obtain access to sensitive resources. However, these studies focus more on trust negotiation policies rather than building trust evaluation approaches. In this work, we organize trust into different trust levels based on trust values and trust trend. The trust value depicts the trust history, while trust trend depicts the future change of trust. Moreover, we apply trust levels to delegation and develop a multi-level delegation model.

4.7 SUMMARY

In this chapter, we propose a multi-level delegation model with trust management, where both delegation tasks and trust are organized into three levels. The delegation task levels are classified according to information sensitivity, while, the trust levels combine trust values and trust trend together to indicate to what extent a delegatee is reliable or trustworthy. Our multi-level delegation model allows that a delegatee in a higher trust level can be assigned with a higher level of task. In the experimental evaluations, we study the precision of our trust model in predicting the trend of the trust and investigate the effectiveness of our proposed multi-level delegation model in terms of information disclosures.

CHAPTER 5

SPECIFYING USAGE CONTROL MODEL WITH OBJECT CONSTRAINT LANGUAGE

This chapter focuses on constraints specification, that is how constraints can be represented. The main contribution of this chapter is to specify constraints of the UCON model with object constraints language (OCL). With OCL, we provide a tool to precisely describe constraints for system designers and administrators. The specification also provides the precise meaning of the new features of UCON, such as the mutability of attributes and the continuity of usage control decisions.

The information in this chapter is based on a published paper [65].

5.1 INTRODUCTION

Developments in information technology, especially in electronic commerce applications, require additional features for access control. The recently proposed usage control (UCON) model is a new access control model that extends traditional access control models in multiple aspects [107] and is considered as the next generation access control model [83]. The usage control (UCON) model was introduced as a unified approach to capture a number of extensions to traditional access control models. In the UCON model, the authorization-based decision process utilizes subject attributes and object attributes. Attributes can be identities, security labels, properties, capabilities, and so on. The UCON model includes obligation and conditions as well as authorizations as part of the usage decision process to provide a richer and finer decision capability. Obligations are requirements that have to be fulfilled for usage

allowance. Conditions are subject and object-independent environmental requirements that have to be satisfied for access. These decision predicates can be evaluated before or during the exercise of a request. In addition, the usage of the target object may require certain updates on subject or object attributes before, during or after a usage exercise.

Park and Sandhu [107, 83] presented the conceptual model of UCON, which consists of several constraints. For example, people may have to click the 'accept' button for license agreement or have to fill out a certain form to download a company's whitepaper. In addition, there are environmental requirements, such as, only IEEE member can access full papers in the IEEE digital library. Constraints can be described in natural languages, such as English, or in more formal languages. Natural language specification has the advantage of ease of comprehension by human beings, but may be prone to ambiguities [108]. Constraints in formal language are suitable for persons with a strong mathematical background, but difficult for average business or system developers to use. For instance, Zhang et al. [116] proposed a formalized specification of the principles of UCON with a temporal logic. The authors in [116] are security experts and system developers who have to understand organizational objectives and articulate major policy decisions.

Although constraints are one of the important components of the UCON model, there is less study in previous works stressing this. This chapter aims to provide a tool to precisely describe constraints for system designers and administrators and specify constraints of the UCON model with object constraints language (OCL). The specification also provides the precise meaning of the new features of the UCON model, such as the mutability of attributes and the continuity of usage control decisions. Another contribution of this chapter is that we give out a formalized specification of the UCON model which is built from these basic constraints, such as authorization predicates, obligation actions and condition requirements.

The rest of this chapter is organized as follows. In Section 5.2, we identify the motivation

of our work in this paper and review the related technologies. Constraints in authorization decisions, obligations and conditions are discussed in Section 5.3. Formalized specifications of usage control model are expressed with OCL in Section 5.4. Some related work is reviewed in Section 5.5 and the summary is provided in Section 5.6.

5.2 MOTIVATION AND RELATED TECHNOLOGIES

Constraints in UCON are one of the most important components that have involved in the principle motivations of usage analysis and design. Using OCL that has been used to express constraints in analysis and design as an industrial standard constraints specification language, we demonstrate that OCL can help us specify previously identified constraints at the system design step.

5.2.1 USAGE CONTROL

UCON has recently received considerable attention as a promising alternative to traditional access control models, such as access matrix [49], mandatory access controls (MAC) [22, 32], discretionary access control (DAC) and role-based access control (RBAC) [37, 84]. Usage control is used for access control in a pervasive environment. There are eight components: subjects, subject attributes, objects, object attributes, rights, authorizations, obligations, and conditions in the usage control model (see Figure 5.1). Subjects and Objects are familiar concepts from traditional access control, and are used in their familiar sense in this paper. A right represents access of a subject to an object, such as read or write. Subject and object attributes are properties that can be used during the access decision process. Examples of subject attributes are identities, group names, roles, memberships, credits, etc. Examples of object attributes are security labels, ownerships, classes, access control lists,

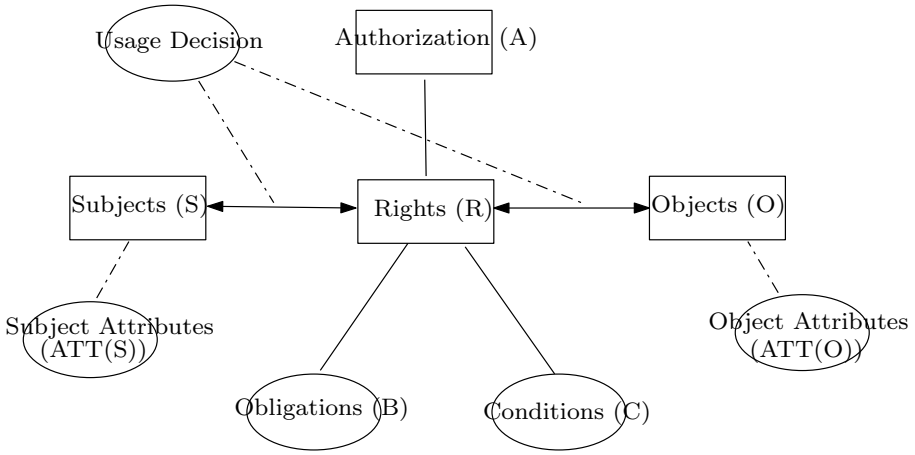


Figure 5.1: Components of UCON model

etc. In an online shop a price could be an object attribute, for instance, a particular e-book may stipulate a 10 price for a 'read' right and a 15 price for a 'print' right.

Authorizations, obligations and conditions are decision factors employed by the usage decision functions to determine whether a subject should be allowed to access an object with a particular right. In addition to these three decision factors, modern information system requires two other important properties called 'continuity' and 'mutability' as shown in Figure 5.2. In traditional access control, authorization is assumed to be done before access is allowed (pre). However, it is quite reasonable to extend this for continuous enforcement by evaluating usage requirements throughout usages (ongoing). the presence of ongoing decisions is called the continuity of UCON. Mutability means that one or more subject or object attribute values can be updated as side-effects of subjects' actions. In the case where attributes are mutable, updates can be done either before (pre), during (ongoing) or after (post) usages shown in Figure 5.2.

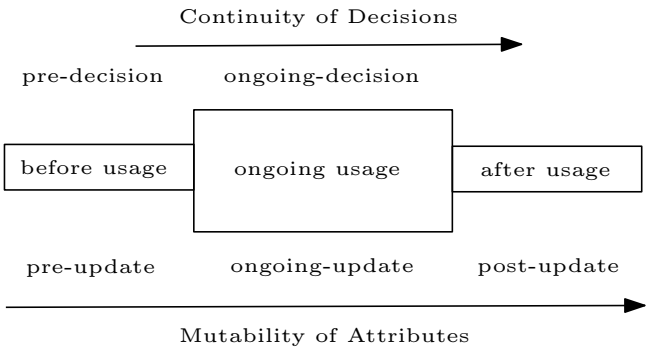


Figure 5.2: Continuity and mutability properties of UCON

5.2.2 UNIFIED MODELING LANGUAGE AND OBJECT CONSTRAINTS LANGUAGE

UML [82] is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It simplifies the complex process of system analysis and design and further software implementation. UML has become a standard modeling language in the filed of software engineering.

OCL expressions are described within the context of an instance of a specific type. In an OCL expression, the reserved word **self** is used to refer to the contextual instance. The type of the context instance of an OCL expression is written with the **context** keyword, followed by the name of the type. The label **invar**: declares the constraint to be an invariant constraint. For example, suppose that employees work for a company and they are involved in projects. These relationships can be modeled using the class model of the UML. If the context is Company, then self refers to an instance of Company. The following shows an example of OCL constraint expression describing a company that has more than 200 employees:

context Company **invar**:

self.employee $\rightarrow$ size $>$ 200

The self.employee is a set of employees that is selected by navigating from Company class to Employee class though an association. The "." stands for a navigation. A property

of a set is accessed by using an arrow "→" followed by the name of the property. A property of the set of employees is expressed using a keyword 'size' in this example.

An OCL expression delivers a subset of a collection. That is, the OCL has special constructs to specify a selection from a specific collection. For example, the following OCL expression specifies that the collection of employees whose age is over 50 is not empty:

context Company **invar**:

self.employee → select(age > 50) → notEmpty

The **select** takes an employee from self.employee and evaluates an expression (age > 50) for the employee. If this evaluation result is true, then the employee is in the result set. More examples can be reviewed in [82].

5.3 CONSTRAINTS IN UCON

Constraints are an important aspect of access control and are a powerful mechanism for laying out a higher-level organizational policy. Consequently the specification of constraints needs to be considered. This issue has received surprisingly little attention in the research literature. Next we will illustrate the main constraints in UCON.

Authorization Constraints

In today's highly dynamic environment, authorizations are predicates based on subject and/or object attributes, which determine whether a subject should be allowed to access an object with particular right. Before authorization, predicates on attributes have to be satisfied. A predicate is a boolean-valued polynomially computable function built from a set of a subject s's and an object o's attributes and constraints. The following examples show how we can specify this type of constraints using OCL.

Example 1*: The subject's credit attribute value in current state of the system should be*

larger than $100.

context State **invar**:

self.attribute $\rightarrow$ subject.credit $>$ $100

Obligation Constraints

In UCON, an obligation is an action that must be performed by a subject before or during an access, such as, filling out a form before playing a licensed music file.

Example 2*: The downloading of a music file may need the requesting subject to click a privacy button.*

context Downloading **invar**:

self.subject.click'privacy' = true

Condition Constraints

Conditions are environmental restrictions that have to be valid before or during a usage process, such as system clock, location, system code, etc.

Example 3*: A subject obtains a permission only when the system clock is in daytime.*

context Obtain a permission **invar**:

self.systemclock.daytime = true

Mutability Constraints

Mutability means that subject and/or object attributes can be updated as the results of an access. There are three kinds of updates: pre_updates, on_updates, and post_updates. The updating of attributes as a side-effect of subject activity is a significant extension of classic access control where the reference monitor mainly enforces existing permissions.

Example 4*: The subject's current usage number of an object is increased by 1 at the time of access and decreased by 1 at the end of access. The update of the object's attribute 'usageNum' can be described as follow:*

context Attributes(o) **invar**:

self.pre_update $\rightarrow$ exists(usageNum = 'usageNum(o)+1')

self.post_update $\rightarrow$ exists(usageNum = 'usageNum(o)$-$1')

5.4 SPECIFYING USAGE CONTROL MODEL WITH OCL

In this section we present a formalized specification of usage control models which is built from the basic components, such as authorization predicates and mutable attributes etc.

5.4.1 $UCON_{preA}$ – PRE-AUTHORIZATION MODELS

Authorizations have been considered as the core of access control and extensively discussed since the beginning of access control discipline. Traditionally, access control research has focused on pre-authorizations in which a usage decision is made before a requested right is exercised. UCON$_{preA}$ models utilize these pre-authorizations for their usage decision processes. In UCON$_{preA}$ models, an authorization decision process is done before usage is allowed. We begin with the UCON$_{preA_0}$ which allows no updates of attributes.

context $preA_0$ **invar**:

init: self.access = 'requesting'

derive: if self.access = 'permitaccess'

then self.$preA(attr(s), attr(o), r)$ = true

else self.access = 'denyaccess'

endif

while $reqesting$ means the access has been generated and is waiting for the system's usage decision, where $preA$ is a functional predicate that utilizes $attr(s)$, $attr(o)$, and right r for usage decision making. We write 'permitaccess' to indicate that subject s is allowed right r to object o. Else, 'denyaccess' indicates that the system rejects the access request.

The UCON$_{preA_1}$ model is similar to UCON$_{preA_0}$ except it takes pre_update attributes into account, i.e., let 'attr:string = self.pre_update $\rightarrow$ size()' in the expression. We use the $size$ property on the set of pre_updated attributes in $preA_1$, where 'size()$\geqslant 1$' indicates that at least subject's or object's attributes are pre_updated.

context $preA_1$ **invar**:

init: self.access = 'requesting'

derive: let attr:string = self.pre_update $\rightarrow$ size() in

if self.access = 'permitaccess'

then

self.$preA(attr(s), attr(o), r)$ = true and

self.pre_update $\rightarrow$ size()$\geqslant 1$

else self.access = 'denyaccess'

endif

The specification of UCON$_{preA_3}$ is similar to that in UCON$_{preA_0}$ except it adds post_update processes, i.e., let 'attr:string = self.post_update $\rightarrow$ size()' in the expression.

***Example 5**: In a DRM pay-per-use application, a $read$ access can be approved when the user Alice's $credit$ is more than an ebook's $value$. Before the access can be begin, an update to Alice's $credit$ is performed.*

context $preA_1$ **invar**:

init: self.access = 'application'

derive: let attr:string = self.pre_update $\rightarrow$ size() in

if self.access = 'read'

then self.attribute $\rightarrow$ $Alice.credit \geq ebook.value$ and

self.pre_update $\rightarrow$ exists ($credit = Alice.credit - ebook.value \rightarrow$ size()$\geqslant 1$)

else self.access = 'denyaccess'

endif

5.4.2 $UCON_{onA}$ – ONGOING-AUTHORIZATION MODELS

In $UCON_{onA}$ model, ongoing-authorizations have been seldom discussed in access control literature. By utilizing ongoing-authorization, monitoring is actively involved in usage decisions while a requested right is exercised. This kind of continuous control is useful for relatively long-lived usage rights. In $UCON_{onA}$, there are four detailed models. $UCON_{onA_0}$ is an immutable ongoing-authorization model that has no update procedure included. $UCON_{onA_1}$ is an ongoing-authorization model with optional pre-updates. UCON$_{onA_2}$ and UCON$_{onA_3}$ include ongoing updates and post updates respectively.

context onA_0 **invar**:

init: self.access = 'accessing'

derive: if self.$onA(attr(s), attr(o), r)$ = false

then self.access = 'revokeaccess'

else self.access = 'endaccess'

endif

$UCON_{onA}$ model introduces an onA predicate instead of $preA$. Since there is no pre-authorization , the requested access is always allowed. The 'accessing' means that the system has permitted the access and the subject has accessed the object immediately after that. In case certain attributes are changed and requirements are no long satisfied, a 'revoke' procedure is performed. We write 'revokeaccess' to indicate that right r of subject s to object o is revoked and the ongoing access terminated. Else, 'endaccess' indicates that a subject finishes the usage and ends the access.

The expressions of onA_1, onA_2 and onA_3 are similar to that in onA_0 except they add the updating of attributes in the expression, i.e., pre_update, on_update, post_update, respectively.

Here, for simplicity we describe onA_1 as follows.

context onA_1 **invar**:

init: self.access $=$ 'accessing'

derive: let attr:string $=$ self.pre_update $\rightarrow$ size() in

if self.$onA(attr(s), attr(o), r) =$ false

then

self.access $=$ 'revokeaccess' and

self.pre_update $\rightarrow$ size()$\geqslant 1$

else self.access $=$ 'endaccess'

endif

Example 6*: Considering a limited number of simultaneous usages, each new access request must be granted and there is only one access generated from a single user at any time. When a new request is generated, one existing user's ongoing access is revoked based on the longest idle time. The policy can be specified as a combination policy of* onA_1*,* onA_2 *and* onA_3 *as follows.*

context onA **invar**:

let attr:string $=$ self.update $\rightarrow$ size() in

init: self.access $=$ 'permitaccess'

derive:

(1) self.pre_update $\rightarrow$ exists (accessingS $= accessingS(o) \cup \{s\}$)

self.pre_update $\rightarrow$ exists (idleTime $= 0$)

(2) if self.access $=$ 'accessing $\wedge$ idle' then

self.on_update $\rightarrow$ exists (idleTime $= idleTime(s) + 1$)

(3) if self.attributes $\rightarrow$ subject.startTime= $Max_{idleTime}(object.accessingS)$

then self.access $=$ 'revokeaccess' and

self.post_update → exists (accessingS = $accessingS(o) - \{s\}$)

endif

where $Max_{idleTime}(object.accessingS)$ is the largest $idleTime$ in the object's $accessingS$ attribute. The first description is an onA_1 rule specifying that whenever a subject tries to access the object, there must be two pre-updates before the subject starts to access. The second rule indicates the mutability of the subject attribute by saying that there must be a continuous update of $idleTime$ whenever the status of subject is idle. The third rule specifies the revocation is determined by the $idleTime$, and the attribute $accessingS$ is updated by removing the subject.

5.4.3 $UCON_{preB}$ – PRE-OBLIGATIONS MODELS

$UCON_{preB}$ introduces pre-obligations that have to be fulfilled at the time of a request and before access is allowed. The $UCON_{preB}$ models consist of two steps. The first step is to select required obligation elements for the requested usage. The $getPreOBL$ function represents the pre-obligations required for s to gain r access to o. Second step is to evaluate whether the selected obligation elements have been fulfilled without any error (e.g., invalid e-mail addresses). The $preFulfilled$ predicate tells us if each of the required obligations is true. In $UCON_{preB}$ models, a request may require multiple pre-obligation elements to be fulfilled. Suppose the set of pre-obligation elements is indicated by M which is based on requests that consist of s, o and r.

context $preB_0$ **invar**:

let M: Set= $\{getPreOBL(s, o, r)\}$ in

init: self.access = 'requesting'

derive: if self.access = 'permitaccess'

then M → select($m|self.preFulfilled = false$) → is empty

else self.access = 'denyaccess'

endif

The expression (M $\rightarrow$ select($m|self.preFulfilled = false$) $\rightarrow$ is empty) indicates that all the required pre-obligation elements are fulfilled by using $preFulfilled$.

The specification of UCON$_{preB_1}$ is similar to that in UCON$_{preB_0}$ except that an pre_update action must be performed before 'permitaccess', i.e., let 'attr:string = self.pre_update $\rightarrow$ size()' in the expression.

context $preB_1$ **invar**:

let M: Set= $\{getPreOBL(s, o, r)\}$ in

init: self.access = 'requesting'

derive:

let attr:string = self.pre_update $\rightarrow$ size() in

if self.access = 'permitaccess'

then M $\rightarrow$ select($m|self.preFulfilled = false$) $\rightarrow$ is empty and

self.pre_update $\rightarrow$ size()$\geqslant 1$

else self.access = 'denyaccess'

endif

The UCON$_{preB_3}$ model is similar to UCON$_{preB_0}$ except it adds post_update processes.

***Example 7**: In an online electronic marketing system, in order to place an order, a customer has to click a button to agree to the order policies. We define an action $click\_agreement$ as an obligation for each other, where the obligation subject is the same as the ordering subject, and the $agree\_statement$ is the obligation object. A customer's $orderList$ is updated by adding the ordered item after he/she places an order. This can be expressed with a $preB_3$ policy as the following.*

context $preB_3$ **invar**:

let M: Set= $\{(s, agree\_statement, order)\}$ in

init: self.access = 'requesting'

derive: if self.access = 'permitaccess'

then M $\rightarrow$ select($m|self.preFulfilled = false$) $\rightarrow$ is empty

self.post_update $\rightarrow$ exists ($orderList = orderList(s) \cup \{o\}$)

else self.access = 'denyaccess'

endif

5.4.4 $UCON_{onB}$ – ONGOING-OBLIGATIONS MODELS

$UCON_{onB}$ models are similar to $UCON_{preB}$ models except that obligations have to be fulfilled while rights are exercised. Ongoing-obligations may have to be fulfilled periodically or continuously. In $UCON_{onB}$ models, there are four detailed models based on mutability issues. $UCON_{onB_0}$ includes an ongoing-obligation predicate instead of a pre-obligations predicate. $UCON_{onB_1}$, $UCON_{onB_2}$, and $UCON_{onB_3}$ are the same as $UCON_{onB_0}$ except that they add pre-updates, ongoing-updates, and post-updates, respectively.

context onB_0 **invar**:

let M: Set= $\{getOnOBL(s, o, r)\}$ in

init: self.access = 'accessing'

derive:

if M $\rightarrow$ select($m|self.onFulfilled = false$) $\rightarrow$ notempty

then self.access = 'revokeaccess'

else self.access = 'endaccess'

endif

Similar to $preB$, the set M shows the selection of required ongoing-obligation elements. The specification (M $\rightarrow$ select($m|self.onFulfilled = false$) $\rightarrow$ notempty) indicates that

not all required ongoing-obligation elements are fulfilled by using $onFulfilled$.

The expressions of onB_1, onB_2 and onB_3 are similar to that in onB_0 except that they add the updating of attributes in the expression, i.e., pre_update, on_update, post_update, respectively. Next, the description of onB_1 is given for simplicity.

context onB_1 **invar**:

let M: Set= $\{getOnOBL(s, o, r)\}$ in

init: self.access = 'accessing'

derive:

let attr:string = self.pre_update $\rightarrow$ size() in

if M $\rightarrow$ select($m|self.onFulfilled = false$) $\rightarrow$ notempty

then self.access = 'revokeaccess' and

self.pre_update $\rightarrow$ size()$\geqslant 1$

else self.access = 'endaccess'

endif

5.4.5 $UCON_{preC}$ – PRE-CONDITIONS MODEL

Conditions are environmental restrictions that have to be satisfied for usages. By utilizing conditions in usage decision process, $UCON_{preC}$ can provide fine-grained controls on usage. Unlike authorization and obligation models, condition models cannot be mutable. $UCON_{preC}$ introduces a pre-conditions predicate that has to be evaluated before the requested rights are exercised.

context $preC_0$ **invar**:

let M: Set= $\{getPreCON(s, o, r)\}$ in

init: self.access = 'requesting'

derive: if self.access = 'permitaccess', then

M $\rightarrow$ select($m|self.preConChecked = false$) $\rightarrow$ is empty

else self.access $=$ 'denyaccess'

endif

In this specification, a set of relevant condition elements M is selected based on a request possibly using subject or object attributes. To allow a request, all of the selected condition restrictions have to be evaluated by using $preConChecked$.

5.4.6 $UCON_{onC}$ – ONGOING-CONDITIONS MODEL

In many cases, environmental restrictions have to be satisfied while rights are in active use. This could be supported within the $UCON_{onC}$ model. In $UCON_{onC}$, usages are allowed without any decision process at the time of requests. However, there is an ongoing-condition predicate to check certain environmental statuses repeatedly throughout the usages.

context onC_0 **invar**:

let M: Set$= \{getOnCON(s, o, r)\}$ in

init: self.access $=$ 'accessing'

derive:

if M $\rightarrow$ select($m|self.onConChecked = false$) $\rightarrow$ notempty

then self.access $=$ 'revokeaccess'

else self.access $=$ 'endaccess'

endif

5.5 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, which have

been discussed in the introduction. The other approach is about the research of temporal access control models, which introduce the temporal attributes into traditional access control with temporal logic. A temporal authorization model for database management systems was first proposed by Bertino et al. [23, 24, 25]. In this model, a subject has permissions on an object during some time intervals or a subject's permission is temporally dependent on an authorization rule. For example, a subject can access a file only for one week. Our specified model is different: we consider the temporal characteristics in a single-usage period, with mutable attributes of subject and object before, during, and after an access, that is, the temporal properties are the result of the mutability of subject and object attributes, which change due to the side effects of access and usages.

Joshi et al. [52] presented a generalized temporal RBAC model (GTRBAC) to specify temporal constraints in role activation, user-role assignment, and role-permission assignment. For example, a user can only activate a role for a particular duration. The concept of temporal constraint is different from the mutability constraints of UCON, since it does not have update actions. The dependency constraint in GTRBAC [51] is similar to the concept of obligation in UCON, but the dependency is more like the implication relation between events in GTRBAC, i.e., if an event happens, it triggers another event; while in UCON, obligations are explicitly required actions to permit an access.

Bettini et al. [26, 27] presented concepts of provisions and obligation in policy management: provisions are conditions or actions performed by a subject before an authorization decision, while obligations are conditions or actions performed after an access. In this work, we distinguish between conditions and obligations. All the actions that a subject has to perform before usage are regarded as obligations, while for future actions, we consider them as the obligations for future usage requests or long-term obligations. Chomicki and Lobo [28] investigate the conflicts and constraints of historical actions in policies. In their paper,

actions are applications activities and constraints are expressed with linear-time temporal connectors. In our paper, we specify obligations as actions required by an access and give formal specification with OCL.

5.6 SUMMARY

This work has discussed the constraints in the UCON model and provide various kinds of constraints representated with object constraint language. We have analyzed the constraints in UCON such as decision actor constraints and mutability constraints etc. We also provide a tool to precisely describe constraints for system designers and administrators. Furthermore, we give out a formalized specification of the UCON model which is built from these basic constraints, such as authorization predicates, obligation actions and condition requirements etc. We show the flexibility and expressive capability of this model by specifying the core models of UCON with extensive examples.

CHAPTER 6

PRIVACY-AWARE ACCESS CONTROL WITH GENERALIZATION BOUNDARIES

In this chapter, we devise a generalization boundary technique to balance privacy and information utilization, satisfying the requirements of both data providers and data users. We propose a privacy-aware access control model, where formalized authorizations are defined relating the permissible usage and specific generalization levels. Compared with traditional access models, our access control model supports a much finer level of control based on "how much information can be accessed for a certain user". Trust-based decision and ongoing access control mechanisms are designed to manage a valid access process at the pre-access and ongoing-access stages, respectively. Finally, we describe the state transition architecture of the privacy-aware access control model to demonstrate how the model works in practice.

The information in this chapter is based on a published paper [64].

6.1 INTRODUCTION

While current information technology enables people to carry out their business virtually at any time in any place, it also provides the capability to store various types of information that users reveal during their activities. Privacy concerns are fueled by an ever increasing list of privacy violations, ranging from privacy accidents to illegal actions. Individuals are becoming more reluctant to carry out business and transactions online potentially leading to many enterprises losing a considerable amount of their profits. Also, enterprises that collect information about individuals are in effect obligated to keep the collected information private and

are required to control the use of such information. Thus, information stored in the databases of an enterprise is not only a valuable property of the enterprise, but also a costly responsibility. By demonstrating good privacy practices, many enterprises try to utilize information analysis and knowledge extraction to provide better services to individuals without violating individual privacy. Changes in legislation around the world and growing consumer attention have changed attitudes towards security and privacy concerns for database systems. This coincides with a substantial body of research on approaches for managing the negotiation of personal information among customers and enterprises [96, 4, 90].

At the heart of protecting privacy is the principle of transparency. Transparency means that when enterprises store data about customers they should disclose to customers what data is being collected and how it is to be used; i.e. for what purpose data is being used and how it is maintained. Starting from the landmark proposal for Hippocratic databases [3], most privacy-aware technologies use purpose as a central concept around which privacy protection is built. Byun and Bertino [20] proposed a model based on a typical life-cycle of data concerning individuals. The use of data generalization[1] helps to significantly increase the comfort level of the data providers. For example, many individuals may not be comfortable with their date of birth being used. Suppose the enterprise promises its customers that this information will be used only in a generalized form; e.g. (08/20/1980) will be generalized to a less specific value (08/1980). This assurance can provide much comfort to many customers and the ability to limit the level of allowed generalization could be valuable in terms of privacy. However over-generalization of data could make it useless; for instance, when address information, such as 14 Regent Street, Toowoomba, Queensland, Australia, is used for some specific data analysis tasks in relation to States in Australia, then the state "Queensland" should be the maximal allowed generalization value. Therefore, the address information

[1] Data generalization refers to techniques that "replace a value with a less specific but semantically consistent value."

generalized beyond the state could be useless. Hence the issue is how to determine whether or not a certain generalization strategy provides a sufficient level of privacy and usability.

Technologies that can provide adequate solutions to this problem require a delicate balance between an individual's privacy and data usability by enterprises. An important component of database management systems that can help to address the above problem is an access control model. Traditional access models, such as Mandatory Access Control (MAC), Discretionary Access Control (DAC), and Role Based Access Control (RBAC) [86, 37], are fundamentally inadequate in this respect. We believe that a new generation of privacy-aware access control models that maximize data usability while minimizing disclosure of privacy is needed. We are facing three challenges in building such a privacy-aware access control model. The first challenge is that the comfort level of privacy varies from individual to individual, and this requires fine-grained access control incorporating generalization techniques with sufficient levels of privacy and usability. The second challenge is that privacy-oriented access control models are mainly concerned with what data object is being used for what purpose(s) rather than which user is performing what action on what data object as in the traditional access control models. The third challenge is how to make the access control technology in a trustworthy fashion, when the data provider and the requester are unknown to each other.

The rest of this chapter is organized as follows. In Section 6.2, we identify the motivation of our work in this paper. We propose the privacy-aware access control model in Section 6.3 and discuss the access control process in Section 6.4. In Section 6.5, we illustrate the state transitions. We show our experimental results in Section 6.6 and provide a brief survey of related work in Section 6.7. Finally, we summary this chapter in Section 6.8.

6.2 MOTIVATION

Following [20], the actual data items[2] are preprocessed before being stored. The pre-processing takes the following form: Each data item is generalized and stored according to a multilevel organization, where each level corresponds to a specific privacy level. Intuitively, data for a higher privacy level requires a higher degree of generalization. Let us briefly first describe the terminologies used in this process:

- *Data Provider*: Data provider refers to the subject to whom the stored data is related. We denote S as the set of data providers.

- *Data Users*: Data users are individuals who access or receive data. Data users are required in a privacy context, as privacy policies will depend on the relationship between the individual requesting data and the individual to whom the data is related to. For example, one type of data users might be $physician$ while another might be $primary\ care\ physician$. We denote U as the set of data users.

- *Privilege*: Some privacy policies make distinctions about who can perform activities based on the action being performed. For example, a policy might state that anyone in the company can $create$ a customer record, but that only certain data users are allowed to $read$ that record. We denote $Priv$ as the set of privileges.

- *Purpose*: Data access requests are made for a specific purpose or purposes. This represents how the data is going to be used by the recipient. For example, the data may be used for $Marketing$ or $Delivery$ purposes. We denote P as the set of purposes.

- *Generalization Level*: Generalization level refers to what extent the data items have been generalized. We denote GL as the set of private levels, which consists of Low, Medium,

[2]Data item refers to the type of data being collected (i.e., attributes), such as $Name$, $Address$. In this paper, we denote D as the set of data items.

Name		Address		Income		Admin	Marketing	Delivery
L	Alice Park	L	123 First St.,Seattle,WA	L	45,000			
M	A. Park	M	Seattle,WA	M	40K-60K	{L,M,H}	{M,H,H}	{M,M,M}
H	A.P.	H	WA	H	Under 100K			

Table 6.1: Privacy information and Metadata

Name	Address	Income	Delivery
A. Park	Seattle,WA	40K-60K	{M,M,M}

Table 6.2: Private information for $Delivery$ purpose

High, and Maximal generalization level, denoted as L, M, H and ML. For example, a Low generalization level on $Address$ means that the address information can be used without any modification.

Table 6.1 illustrates some fractional records and privacy requirements stored in a conceptual database relation. Note that each data item is stored at three different privacy levels, $Low, Medium, High$. Take the $address$ data as an example: the entire address is regarded as Low, city and state are at $Medium$ and state at $High$. Admin and Marketing are metadata columns storing the set of privacy levels of data for $Admin$ and $Marketing$ purposes respectively. Further, a data provider submits his/her privacy requirements, which specify permissible usages of each data item and a level of privacy for each usage. For instance, {M, H, H} under Marketing indicates that for the $Marketing$ purpose data users can only access $Name$ at the $Medium$ privacy level while accessing $Address$ and $Income$ at the $High$ level.

We can see that the access to each data item is strictly governed by the data provider's requirements. Before data access, authorizations on each data item have already been granted through the permissible usage requirements. However, different people may have different feelings about their information being used for some purposes. For instance, some consumers may feel that it is acceptable to disclose their purchase history or browsing habits in return for better services; others may feel that revealing such information violates their

privacy. Differences in individuals suggest that access control models should be able to maximize information utility, which may be neglected by data providers although wanted by data users. For example, if a data provider selects {M, M, M} on $Name$, $Address$, $Income$ for $Delivery$ purpose, (i.e., the data user has been authorized to access $Name$, $Address$, and $Income$ only in medium level shown in Table 6.2) then, the information could be useless for the data user who wants to fulfill the delivery purpose because full name and address are necessary information for delivery. Further, the {M, M, M} selection may increase the chance of disclosure of the unnecessary information $Income$ since the more people who know, the more likely it would be disclosed. Authorizations incurred by this selection could not protect data privacy (e.g., $Income$, to some degree) nor maintain data usability.

To solve this problem, we need metrics that methodologically measure the privacy and usability of generalized data. It is necessary to devise efficient generalization techniques that satisfy the requirements of both data providers and data users. In this paper, we propose a privacy-aware access control model with generalization boundaries, which can maximize data usability and minimize privacy disclosure. In particular, we

• Formalize the authorizations with the specific purpose and generalization levels specified on each data item and investigate two other factors, obligations and conditions.

• Propose a trust-based decision policy with trust evaluation techniques to handle access security with regard to a requester's trust before data access and design authorization and access functions to handle access security with regard to the retention period and generalization level during data accessing.

• Study the state transition of our proposed privacy-aware access control model and illustrate how the model works in practice.

• Evaluate our proposed access control model on both real-life and synthetic data sets to show its efficiency and effectiveness.

6.3 PRIVACY-AWARE ACCESS CONTROL MODEL

By using data generalization, data providers can specify their privacy requirements using a privacy level for each data item. Data for a higher privacy level requires a higher degree of generalization; i.e., each privacy level is accompanied with a generalization level. However, over-generalized data may render data of little value or useless. In this section, we introduce a privacy-aware access control model with generalization boundaries.

6.3.1 GENERALIZATION BOUNDARY

In order to specify a generalization boundary, we introduce the concept of a maximum allowed generalization level that is associated with each data item. This concept is used to express to what extent the data user thinks the data item could be generalized, such that the resultant generalized data item would still be useful. Limiting the level of generalization for the data item is necessary for various usage of the data. For instance, when data related to Australian states is used for some specific analysis tasks, the data user will select the level corresponding to the states as the maximal allowed generalization level. Address information generalized beyond the Australia state level could be useless. In this case, the only solution would be to ask the data provider to make a decreased level of generalization until the generalized data satisfies the maximum allowed generalization level requirement (i.e., no address is generalized further than the Australian state).

Definition 6.1. *Let D be the set of data attributes and P be the set of purposes. For each data attribute $d \in D$ and purpose $p \in P$, the* ***maximum allowed generalization level*** *of d under purpose p, denoted by $MAGLel(d, p)$, satisfies that the data attribute d is permitted to be generalized only up to $MAGLel(d, p)$.*

Name		Address		Income		Delivery
L	Alice Park	L	123 First St.,Seattle,WA	L	45,000	{MAGLel(Name, Delivery),
M	A. Park	M	Seattle,WA	M	40K-60K	MAGLel(Address, Delivery),
H	A.P.	H	WA	H	Under 100K	ML}
ML	*	ML	*	ML	*	

Table 6.3: Generalization boundaries for $Delivery$ purpose

Name	Address	Income	Delivery
Alice Park	123 First St., Seattle,WA	*	{L,L,ML}

Table 6.4: Ideal information for $Delivery$ purpose

We assume that the generalization level is equal to the privacy level in this paper. The maximal generalization level, denoted ML, corresponds to generalizing a data value to $*$. For simplicity of discussion, we only consider the generalization levels: low (L), medium (M), high (H) and (ML). For example, if D = {Name, Address, Income}, P = {Admin, Marking, Delivery}, then we can define the maximum allowed generalization level of $Name$ under purpose $Delivery$, $MAGLel(Name, Delivery) = L$.

Note that the maximum allowed generalization level of the data could be different for different purposes. For example, the maximum allowed generalization level of $Address$ could be Low for $Delivery$ purpose, whereas it may be $High$ for $Marketing$ purpose. Usually, for a certain purpose, the data user only has generalization restrictions for some necessary data items; e.g., there should be restrictions on $Name$ and $Address$ for $Delivery$ purpose but no restrictions on $Income$. If for a particular data item there are no any restrictions with respect to its generalization, then the maximal generalization level ML is specified for the usage of this data. In this case, the requirement of providing sufficient privacy and usability is satisfied by the following description.

Definition 6.2. *Let P be the set of access purposes and D be the set of data items, for each purpose $p \in P$, the set $N_p \subseteq D$ denotes all necessary data attributes to fulfill the purpose p. The **privacy-aware generalization boundaries** for p satisfies the following:*

- *for* $\forall d \in N_p$*, the data attribute* d *is permitted to be generalized only up to* $MAGLel(d, p)$*;*
- *for* $\forall d \notin N_p$ *and* $d \in D$*, the data attribute* d *is permitted to be generalized up to* ML*.*

For instance, if D = {Name, Address, Income} and P = {Admin, Marking, Delivery}, then since the full name and address are necessary to fulfill the $Delivery$ purpose, $N_{Delivery}$ = {Name, Address}. Table 6.3 shows the example of privacy-aware generalization boundaries for the $Delivery$ purpose. Because of $Name, Address \in N_{Delivery}$, the generalizations on $Name$ and $Address$ are only permitted up to $MAGLel(Name, Delivery)$ and $MAGLel(Address, Delivery)$ (i.e., Low and Low), respectively. On the other hand, for $Income$, there are no requirements with respect to its generalization, since $Income \notin N_{Delivery}$, so the maximal generalization level ML is specified for the usage of $Income$. The information obtained by the data user is shown in Table 6.4.

The above example shows that our proposed generalization boundary strategy can maximize data usability while, at the same time, minimizing disclosure of data privacy. Moreover, the specific generalization boundaries actually describe the permissible usage of each data item, and the permissible usage further grants the data user to access each data item from a specific generalization level. Such a finer level access control could satisfy the requirements of both data providers and data users. Now the issue is how to build a formal access control model with specific generalization boundaries that can balance data privacy and usability. We discuss this question in detail in the next section.

6.3.2 PRIVACY-AWARE AUTHORIZATIONS

Authorization is the act of checking to see if a data user has the proper permission to access the particular data or perform a particular action. In addition to the traditional authorization factors, data items, data users and privileges, all authorizations in this paper are extended to

include the specific purpose and generalization level on each data item.

Definition 6.3. *A generalized authorization is a 5-tuple* $(u, d, priv, p, gl)$*, where* $u \in U$*,* $d \in D$*,* $priv \in Priv$*,* $p \in P$*,* $gl \in GL$*.*

As previously mentioned, D is the set of data items. The tuple $(u, d, priv, p, gl)$ states that the data user u has been authorized to perform $priv$ on the data item d under generalization level gl for purpose p. For example, the tuple $(Tom, address, access, delivery, L)$ denotes that Tom was authorized with privilege $access$ of the customer's $address$ at Low generalization level for the $delivery$ purpose.

Moreover, personal information is retained only as long as necessary for the fulfillment of the purpose for which it has been collected. Retention period refers to how long the information is stored. For example, if the retention period for $Name$ is one month, the name information can only be retained for one month. We use time intervals to describe retention period, e.g., [12/02/2008, 12/03/2008]. We denote T as the set of time intervals. If a certain data item was collected for a set of purposes, it is kept for the limited retention period of the purpose. We refer to an authorization together with its usage time as a temporal generalized authorization. A time interval is also associated with each authorization, imposing lower and upper bounds to the potential usage.

Definition 6.4. *A temporal generalized authorization is a 6-tuple* $(t, u, d, priv, p, gl)$*, where* $t \in T$*,* $u \in U$*,* $d \in D$*,* $priv \in Priv$*,* $p \in P$*,* $gl \in GL$*.*

A tuple $([t_a, t_b], u, d, priv, p, gl)$ states that the data user u has been authorized to perform $priv$ on the data item d in the generalization level gl for the purpose p in the time interval $[t_a, t_b]$. We denote AU as the set of temporal generalized authorizations and $\sigma_{au}(*)$ as the function used to extract the element(s) $*$ in an authorization $au \in AU$. A temporal generalized authorization $au = ([12/06/2008, 10/08/2008], Tom, income, read, admin, M)$,

means that between June 12, 2008 and August 10, 2008, Tom was authorized the privilege to $read$ the customer's $income$ at the generalization level $Medium$ for the $admin$ purpose. Here, $\sigma_{au}(t)$ refers to the time interval [12/06/2008, 10/08/2008] and $\sigma_{au}(u, d, priv, p)$ returns to the tuple (Tom, income, read, admin).

6.3.3 AUTHORIZATION SPECIFICATION

An authorization is an approval of a particular mode of access to one or more objects in the system. Observe that in a group of authorization assignments, two authorization assignments may interact with each other when they share the same user, same data and same action. Purposes mentioned in an authorization naturally have a hierarchical relationships among them. For instance, a group of purposes such as direct-marketing and third-party marketing can be represented by a more general purpose, marketing. More specific authorizations may deal with more specific purposes that fall under the domain of a high-level purpose. This suggests that purpose can be organized according to the hierarchical relations to simplify their management. Mathematically, a purpose hierarchy is represented as a tree. Each purpose (except the root purpose) has exactly one parent purpose and there are no cycles. A parent node represents a more general purpose than those represented by its children nodes. Thus the hierarchy of purposes can be intended as a grouping of more particular purposes into more general ones. The same argument also could apply to generalization levels. Generalization refers of replacing the actual value of the attribute with a less specific, more general value which is faithful to the original [94, 92, 93]. For example, the name 'Carol Jones' can be generalized to a less specific value 'C. Jones' or further generalized to 'C.J.'. As for purposes hierarchies, a generalization hierarchy is represented as a tree structure. The meaning associated with the generalization hierarchy is analogous to the one mentioned for purpose hierarchies. Here, we use operation " $\geq$ " to indicate the dominance relationship in

the purpose hierarchy and generalization hierarchy.

Explicit(implicit) authorization: The introduction of hierarchies of purpose and generalization level with a retention period lead us to get two types of authorizations, called explicit authorizations and implicit authorizations.

Definition 6.5. *Let* $au_1 = (t_1, u_1, d_1, priv_1, p_1, gl_1)$ *and* $au_2 = (t_2, u_2, d_2, priv_2, p_2, gl_2)$ *be two authorization in* AU. *We say that* au_1 *is an explicit authorization of* au_2 *(or* au_2 *is an implicit authorization of* au_1*) only if one of the following conditions satisfies:*

- $(t_1 \supseteq t_2) \wedge (u_1 = u_2) \wedge (d_1 = d_2) \wedge (priv_1 = priv_2) \wedge (p_1 = p_2) \wedge (gl_1 = gl_2)$
- $(t_1 = t_2) \wedge (u_1 = u_2) \wedge (d_1 = d_2) \wedge (priv_1 = priv_2) \wedge (p_1 \geq p_2) \wedge (gl_1 = gl_2)$
- $(t_1 = t_2) \wedge (u_1 = u_2) \wedge (d_1 = d_2) \wedge (priv_1 = priv_2) \wedge (p_1 = p_2) \wedge (gl_1 \geq gl_2)$

For example, let $au_1, au_2, \ldots, au_9$ be authorizations, where

$au_1 = ([9AM, 5PM], Tom, email, read, Marking, M)$,

$au_2 = ([9AM, 3PM], Tom, email, read, Marking, M)$,

$au_3 = ([9AM, 3PM], Tom, email, read, Third-party\ Marking, M)$

$au_4 = ([9AM, 3PM], Tom, email, read, Third-party\ Marking, H)$,

then they can be represented as a tree.

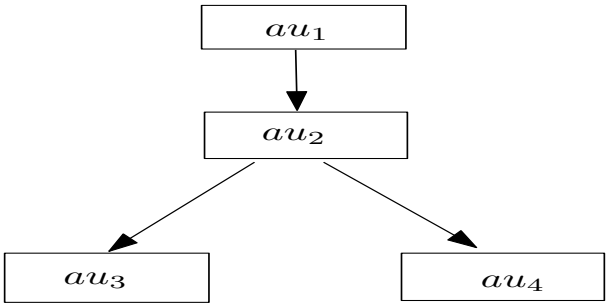


Figure 6.1: Authorization tree

Conflicting authorizations: Complex environments, such as large enterprises, usually have to comply with complex security and privacy policies. As such, it is possible that the more complex a security policy is, the larger the probability that such policy contains inconsistent and conflicting parts is. In particular, authorization assignments could conflict because of new requirements, new regulations, or just human mistakes.

Consider the following authorization assignments:

au_1 =([9AM, 5PM], Bank manager, loan, approve, Marking, Low)

au_2 =([9AM, 5PM], Bank manager, loan, fund, Marking, Low).

Notice that there are different privileges related to the same user working on the same data in the generalization level for the purpose in the time interval. A tricky issue here is that the privileges of approving a loan in a bank and that of funding a loan are conflicting. Therefore, these two authorizations conflict with each other since they have conflicting privileges.

Definition 6.6. *Let* $au_1 = (t_1, u_1, d_1, priv_1, p_1, gl_1)$ *and* $au_2 = (t_2, u_2, d_2, priv_2, p_2, gl_2)$ *be two authorization in* AU. *We say that* au_1 *and* au_2 *are conflicting only if* $priv_1$ *and* $priv_2$ *are conflicting.*

6.4 ACCESS CONTROL PROCESS

After each data is granted with authorizations according to different purposes, an access request is needed to access the data items. In this paper, we assume that each access request is associated with an access time and a specific purpose. It is not trivial for a system to correctly infer the purpose of a query as the system must correctly deduce the actual intention of database users.

Definition 6.7. *An access request is a 5-tuple* $(t, u, d, priv, p)$ *where* $t \in T$ *is the time when the access is requested,* $u \in U$ *is the data user who requires the access,* $d \in D$ *is the data*

item to be accessed, $priv \in Priv$ is a privilege exercised on the data, and $p \in P$ is the purpose for which the data is going to be used.

The tuple $([t_a, t_b], u, d, priv, p)$ states that the data user u requests to perform $priv$ on the data item d for purpose p in the time interval $[t_a, t_b]$. We denote R as the set of access requests and for an access request $r \in R$, $r(*)$ refers to the element(s) $*$ in an access request r. For example, the access purpose $r = ([10/07/2008, 20/07/2008], Tom, income, read, admin)$ means that between July 10, 2008 and July 20, 2008, Tom requests to $read$ the customer's $income$ information for the $admin$ purpose. Here, $r(t)$ refers to the time interval [10/07/2008, 20/07/2008].

Under a request, traditional access process refers to a general way of controlling access to data items and makes authorization decisions based on the identity of the resource requester. Unfortunately, when the resource owner and the requester are unknown to one another, access control based on identity may be ineffective. Access control technology can be used as a starting point for managing personal identifiable information (PII) in a trustworthy fashion. It is important that data items are accessed by persons who are trusted, and this requires that trust-based decisions should be made by data providers according to the data user's trust value. Next, we discuss the management of a valid access process through the trust-based decision policy.

6.4.1 TRUST-BASED DECISION MECHANISM

Trust means the liability and trustworthiness of a trusted agent's behavior. There are two approaches to obtaining an agent's trust: experience by interacting with the agent, and recommendation of other agents [106]. In this paper, we evaluate the trust value in three steps (as show in Figure 6.2): (1) Calculate the trust value based on histories; (2) Calculate the trust value from recommendators; (3) Combine the observed trust values from histories and

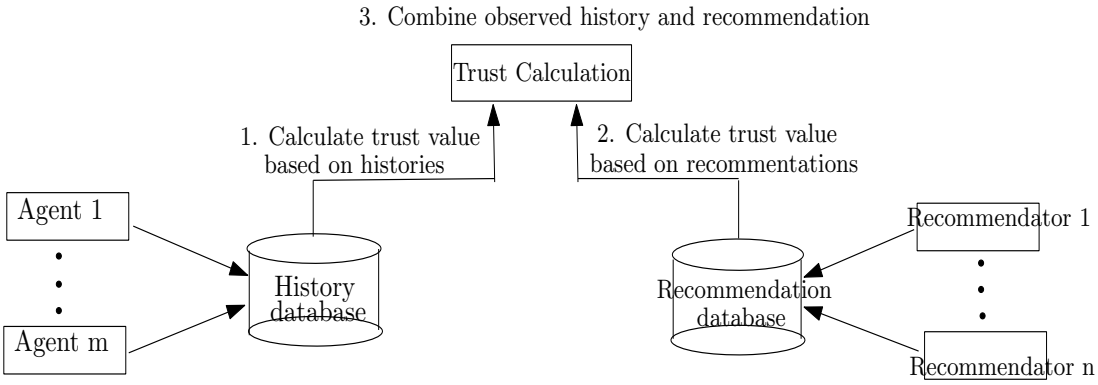


Figure 6.2: Trust evaluation

recommendations.

Step 1: Calculate trust based on histories.

Let m denotes the total number of transactions performed by a data user u during the given period, and $S(u,i)$ denote the satisfaction degree of the participating agent in u's i-th transaction, $S(u,i) \in [0,1]$. If the transaction context factor of u's i-th transaction is $TF(u,i)$, then u's trust can be evaluated by direct experience as follows:

$$T_1(u) = \frac{\sum_{i=1}^{m} S(u,i) \times TF(u,i)}{\sum_{i=1}^{m} TF(u,i)} \tag{6.1}$$

Here, $TF(u,i) \in (0,1)$ is the weight to indicate the influence of a transaction on trust value. If the value of $TF(u,i)$ is large, the transaction has more influence on trust value. Further, if a data user u behaves in a satisfactory manner in all related transactions, i.e. $S(u,i) = 1$ for every i, then u can be regarded as completely trustworthy, i.e. $T(u) = 1$.

Step 2: Calculate trust based on recommendations.

Now we consider the situation of obtaining u's trust from others' recommendation. Let n denote the total number of the recommendations, and $P(u,j) \in (0,1)$ denote the normalized amount of satisfaction of recommendation for data user u in its j-th transaction. $TP(u,j) \in (0,1)$ denotes the weight of j-th transaction, the recommendation-based trust value can be

Time	$S(u,i)$	$TF(u,i)$	$P(u,i)$	$TP(u,i)$
1^{st}	0.8	0.4	0.7	0.3
2^{nd}	0.7	0.6	0.8	0.5
3^{rd}	0.9	0.6	0.8	0.4
4^{th}	0.6	0.8	0.5	0.6
5^{th}	0.9	0.7	0.7	0.8
$T_1(u) = \frac{\sum_{i=1}^{m} S(u,i) \times TF(u,i)}{\sum_{i=1}^{m} TF(u,i)} \approx 0.7$				
$T_2(u) = \frac{\sum_{j=1}^{n} P(u,j) \times TP(u,j)}{\sum_{j=1}^{n} TP(u,j)} \approx 0.6$				
$T(u) = \alpha \times T_1(u) + (1-\alpha) \times T_2(u) \approx 0.68\ (\alpha = 0.6)$				

Table 6.5: Example of trust calculation for data user u

calculated as follows:

$$T_2(u) = \frac{\sum_{j=1}^{n} P(u,j) \times TP(u,j)}{\sum_{j=1}^{n} TP(u,j)} \tag{6.2}$$

Step 3: Merge history-based trust with recommendations.

Now we consider both the trust value from contacting with data user u and the trust value from others' recommendations. Choose a power $\alpha \in (0,1)$, then we can calculate u's trust as follows:

$$T(u) = \alpha \times T_1(u) + (1-\alpha) \times T_2(u) \tag{6.3}$$

The above method for calculating a data user's trust combines the trust information based on the past experiences in interacting with this data user and other's recommendations, and considers the influence of a transaction context. With this approach, we can obtain the data user's trust value, which is assigned in the range $[0,1]$.

Table 6.5 details an example on how to calculate a data user's trust value, where five transaction behaviors are recorded and recommended. The satisfaction degree from participating agents and commentators are given under $S(u,i)$. According to formulas (1) and (2), we can get the trust value $T_1(u) \approx 0.7$ based on histories and $T_2(u) \approx 0.6$ based on recommendations. Combining the two values gives the total trust value $T(u) \approx 0.68$ when the power α is chosen on 0.6.

The data user's trust status is dynamic. An agent who once behaved well might subsequently behave maliciously. So a data user's trust value is only valid for a period of time, and it should be updated timely. Now assume that a data user requests to read a data item. The data accessible to the request normally depends on whether the requester's trust value is higher than the data provider's trust threshold for reading the data. Different accesses or services require participating users with different trust status. For example, a payment service may require that the parties are highly reliable, while ordinary file share service has a lower requirement for an agent's trust. Write access to a file needs a higher trust degree than read access to the same file, and the access to a confidential file requires a higher degree of trust than access to an ordinary file.

In our model, the data provider's trust threshold is defined as the minimum trust value for obtaining operation permission. Access is permitted only when the requester's trust degree is higher than the data provider's trust threshold. Conversely, when a data requester's trust degree is less than the data provider's trust threshold for an operation, the data requester will be prohibited from performing the operation.

The trust-based decision is described as follow:

Let S, U, D, $Priv$ be the set of data providers, data requester (users), data items, and operations. Then

$P_D \subseteq Priv \times D$ denotes the operations on data items

$TT\_S : S \times P_D \rightarrow [0, 1]$ (The data provider's trust threshold for performing an operation on a data item)

$T\_U : U \times P_D \rightarrow [0, 1]$ (A data requester's trust degree for performing an operation on a data item)

$F : S \times U \times P_D \rightarrow \{0, 1\}$ (Trust-based decision)

In a trust-based decision, $F : S \times U \times P_D \rightarrow \{0, 1\}$ denotes a mapping from the data

requester's operation permission on the data item to the set $\{0, 1\}$. Here 1 denotes that access is permitted and 0 denotes that access is denied.

When a data requester requests to perform an operation on a data item, the access control system judges whether the trust degree of the data requester is higher than the data provider's trust threshold or not, and then decides to map the access permission to 0 or 1. That is,

$$\forall s \in S, u \in U, p_d \in P_D$$

$$F(s, u, p_d) = T\_U(u, p_d) \geq TT\_S(s, p_d)$$

If the trust degree of data requester u for performing operation on d is not less than the data provider's trust threshold, the access permission is mapped to 1 and access is permitted; otherwise, access permission is mapped to 0 and access is denied. This can be seen as an instance of the trust enhanced security model and framework recently proposed in [67].

6.4.2 ONGOING ACCESS CONTROL MECHANISM

The above trust-based decision mechanism handles access security before access, but does not consider the authorization of data provider or data items' security sensitivity during the data usage. In the process of access control management, the ongoing access control mechanism is needed in order to achieve an efficient access control management.

As far as an authorization is concerned, the first step is to find all valid authorizations under the request. This is checked by the valid authorization function.

Authorization check function: The valid authorization function is used to judge whether the current authorization au is valid. It can be expressed as follows:

$$G(r) = \begin{cases} au & \text{if } (r(u) = au(u)) \wedge (r(d) = au(d)) \wedge (r(priv) = au(priv)) \\ & \quad \wedge (r(p) = au(p)) \wedge (r(t) \subseteq au(t)) \\ \phi & \text{others} \end{cases}$$

Here $au \in AU$, and $G(r)$ returns a set of valid authorizations. Except for checking the same data user to perform the same privileges on the same data items for the same purpose, the period constraint of an authorization plays an important role. If the request access time is within the retention period, it refers to the authorization as valid, otherwise, the authorization is invalid.

However, a valid authorization function is not enough for an access request, since it only checks whether an authorization exists in the current AU from the angle of the retention period. Besides that, the generalization level decides whether the access of the request is valid according to the current authorizations. Therefore, a valid access function is needed conveniently. Here, we use $r(gl)$ to indicate the generalization level that the request is going to access. If there exists a valid authorization satisfying $r(gl) = au(gl)$ (where $au(gl)$ refers to the generalization level in this authorization) the access is permitted, otherwise, the access is rejected.

Access check function: The valid access function can be expressed as follows:

$$F(r) = \begin{cases} true & \exists au \in G(r), r(gl) = au(gl) \\ false & \text{others} \end{cases}$$

where r is an access request. If $F(r)$ is true, the access is valid. Otherwise, it is invalid.

After a data user submits an access request r and $F(r)$ is true, the user is permitted to access the data. After a data user submits an access request r and $F(r)$ is true, the user is permitted to access the data. During the process of access, there are three kinds of situations that need to be considered. If a requested authorization tuple is a time independent authorization, then the authorization au is invoked. If it is a temporal authorization, when the time exceeds the retention time, the au is illegal. If the data item being accessed is not in the same generalization level, access is rejected. The pseudo code of the ongoing access control policy is described in Algorithm 1.

Algorithm 1: Access control(AU, r)

Input: an access request r and the set of current temporal generalized authorizations AU

Let $G(r) = \{au | au \in AU\}$; /**use the valid authorization function to return a set of authorization tuples, and then judge whether the authorization is valid**/
If $G(r) == \phi$
    return false; /**This authorization does not exist**/
else if $r(t) \nsubseteq au(t)$, for $\forall au \in G(r)$
    return false; /**No legal Authorization**/;
else if
    let $k = F(r)$; /**use the valid access function to return a boolean value, with which to judge whether the access is valid.**/
        if $r(gl) \neq au(gl)$, for $\forall au \in G(r)$
          then $k == false$
            return false; /**The access is rejected'**/
else
        $k == true$
    return true. /**The access is succeeded'**/

6.5 STATE TRANSITIONS

In previous sections, the privacy-aware access control model has been discussed in detail. In this section, the state transition of the proposed access control model is given.

In our privacy-aware access control model, the most important thing is that all authorizations are derived from the permissible usage of each data item. By applying data generalization boundary techniques on each data item, authorizations for a user to access data items in specific generalization levels are specified. Three different attributes are required to meet these authorizations:

- *The time interval*. This includes the start time and end time for which access is permitted. At the end time, the privilege for using data items is revoked.
- *The validity period*. Access to a data item can be permitted only during the valid period of usage.
- *Generalization level*. The data item can only be accessed under the authorized generalization level.

The operations in the authorizations are to grant/revoke privileges to/from data users. Privileges are revoked under the following two situations:

(1) Revocation by time interval if the time interval of authorizations has expired.

(2) Revocation by generalization level if the data item is accessed at the wrong generalization level.

Further, an administrator of the system can make a forced revocation decision. For example, if a security administrator notices that a data user often sends many access requests without using services, the administrator may take actions on this user, such as revoking his authorization to prevent denial of service (DoS) attacks.

In the practical access control process, authorizations are assumed to be done before access is allowed (pre-check). However, it is quite reasonable to extend this for continuous

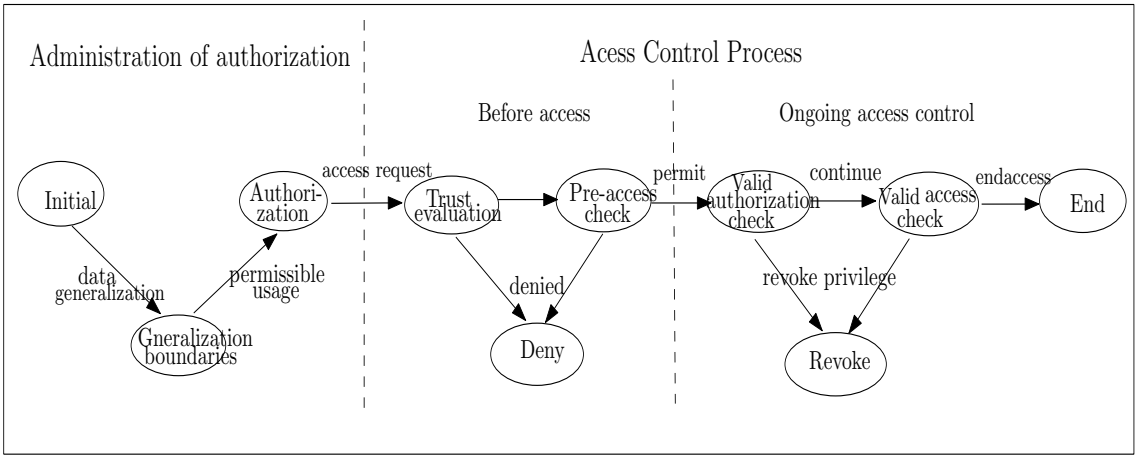


Figure 6.3: The state transition of privacy-aware access control model

enforcement by evaluating usage requirements throughout usage (on-going). The presence of on-going decisions is called the continuity. In the pre-access stage, we need to check whether the requester's trust degree is higher than the data provider's trust threshold and the required obligations and conditions are satisfied. In the ongoing-access stage, we need to check whether the valid authorization and access functions are satisfied. The on-going access may be revoked if the security policies are not satisfied. The pre-access decision policy and ongoing access control policy combined together construct a secure protection system. The state transition of privacy-aware access control actions is given in Figure 6.3. The states and actions in Figure 6.3 are explained bellow.

(1) Initial: the initial state of the metadata.

(2) Data generalization: replacing a data value with a less specific but semantically consistent value.

(3) Generalization boundaries: restricting the maximum allowed generalization level of each data item.

(4) Permissible usage: the type of potential data usage (i.e., purpose).

(5) Authorization: granting privileges of service to data users if data users meet authorization requirements of the system.

(6) Access request: a user request to access digital objects.

(7) Trust evaluation: checking whether the requester is trustworthy or not.

(8) Pre-access check: checking whether the trust threshold is satisfied.

(9) Permitted and denied: if the requester is trustworthy, the access to data items is permitted; otherwise, denied.

(10) Valid authorization check: checking whether the requested access time is in the valid retention period.

(11) Continued and revoked: if the time interval has not expired during the valid period, an access to data items is continued; otherwise, it is revoked.

(12) Valid access check: checking the accessed generalization level of the data item.

(13) Revoke privilege and endaccess: if the data item is accessed at a wrong generalization level, the system will revoke the privileges.

(14) Deny, Revoke and End: three final states. Deny is the state of refusing to access without revoking privileges. Revoke is the state after the action of revoke privileges, while End is the state after the action of endaccess.

From the analysis of state transitions in a privacy-aware access control, it is clear that an access is not a simple action, but consists of a sequence of actions and active tasks.

6.6 EXPERIMENTAL EVALUATIONS

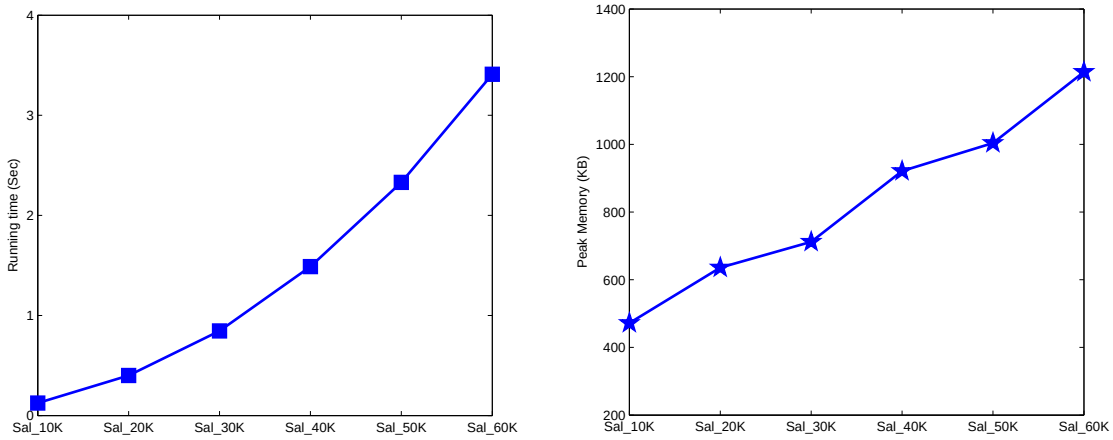
The main goals of the experiments are two-fold. First, we study the performance and storage overheads of our proposed access control model. We consider the impact of the number of attributes accessed and the number of generalization levels on the execution time and storage overheads, and we also examine the scalability of our approach by experimenting with relations of different cardinalities. Second, we investigate the effectiveness of our model in terms of *disclosure rate*, which is a novel metric defined to measure to what extent the access control models can protect sensitive information disclosure.

Experimental setup: We employ two data sets in our experimental evaluations. For evaluating the performance and storage overheads, we adopt a real-world data set CENSUS, downloadable at `http://www.ipums.org`, which contains the personal information of 500K American adults. The data set has 9 discrete attributes summarized in Table 6.6. From CENSUS, we create one set of micro tables, in order to examine the influence of dimensionality and the impact of cardinality. The set has 6 tables, denoted as SAL-10K, $\cdots$, SAL-60K, respectively. Specifically, SAL-n ($10K \leq n \leq 60K$) indicates the number of records randomly sampled from CENSUS data set, and each record consists of nine attributes shown in Table 6.6. We evaluate the execution time of our approach by varying the cardinality of the data sets, the number of attributes and the number of generalization levels. We adopt the peak memory to measure the storage overheads, which indicates the most memory used during the implementation of our method.

To evaluate the effectiveness of our proposed access control model, we generate a synthetic data set with 50K records, and each record contains 1000 numeric attributes with the values randomly chosen from [0,1]. In this set of experiments, we set the number of generalization levels to be three, being High(H), Medium(M) and Low(L). For the implementation, we vary the portion of the attributes with different access levels and investigate their impact

Attribute	Number of distinct values
Age	78
Gender	2
Education	17
Marital	6
Race	9
Work-class	8
Country	83
Occupation	50
Salary-class	50

Table 6.6: Summary of attributes

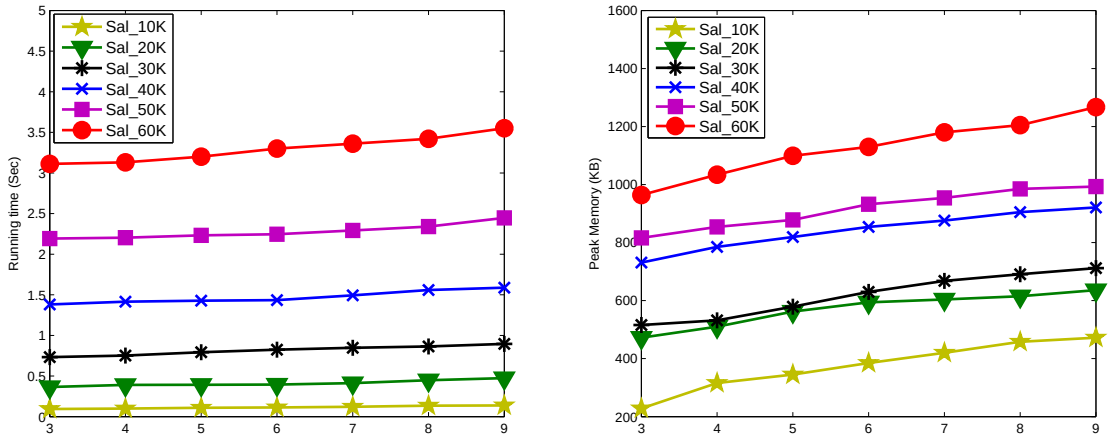


(a) (b)

Figure 6.4: Time and space complexity when data percentage varies; (a) The running time when data percentage varies; (b) Peak memory when data percentage varies.

on the effective measurement *disclosure rate*. In order to reduce randomness, we run each test 500 times for each data and use the average to mark the graph.

Performance: Figures 6.4(a) and 6.5(a) show the computation overhead of our proposed privacy aware access control model with generalization boundaries. In this set of experiments, the computation is run through data sets SAL-10K, $\cdots$, SAL-60K, the default number of attributes accessed is nine, and the generalization hierarchy is set to have three levels. As shown in Figure 6.4(a), the computation overhead increases as the number of records grows.

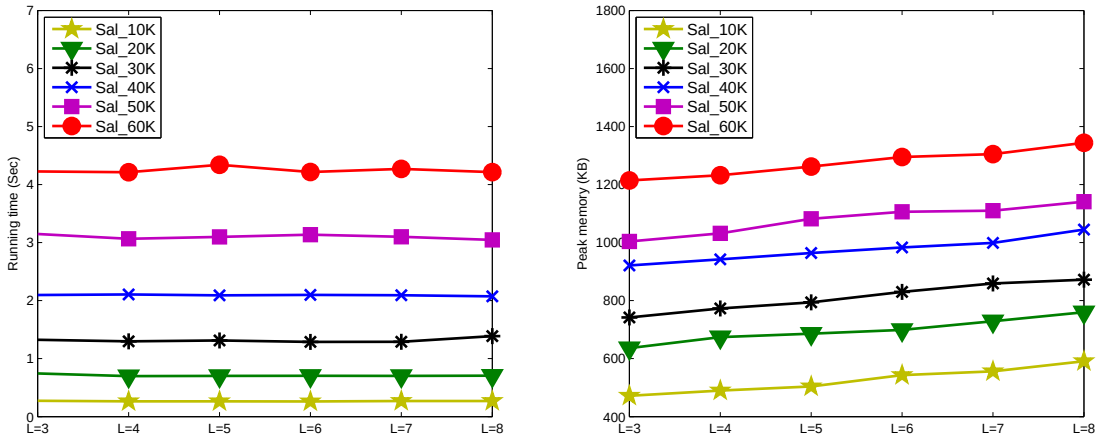


(a) (b)

Figure 6.5: Time and space complexity when the number of attributes varies; (a) The running time when the number of attributes varies; (b) Peak memory when the number of attributes varies.

As expected, the performance becomes poorer as the cardinality of the data set increases. Figure 6.5(a) plots the effect of the number of attributes on the execution time. The result is expected since the cost of computing increases with more dimensions. Figure 6.6(a) describes the effect of the number of generalization levels on the computation overhead. From the figure, we can see that the running time is almost steady while varying the number of levels in the generalization hierarchy.

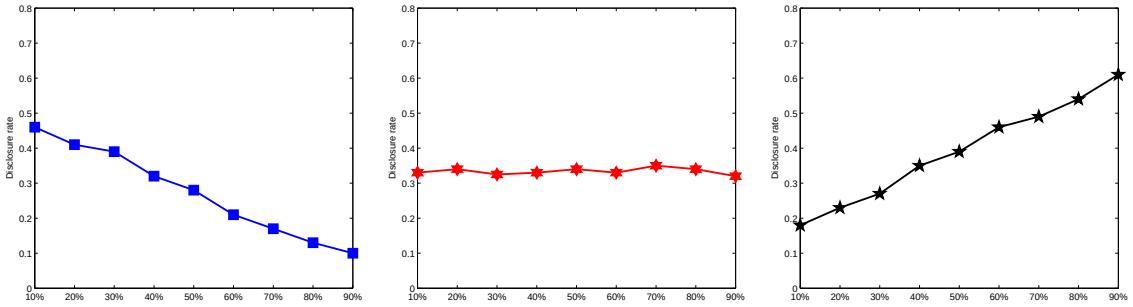
Storage overhead: Figures 6.4(b) and 6.5(b) display the space overhead of our proposed access control model. As shown in Figures 6.4(b) and 6.5(b), the storage overhead increases when the number of records grows and the number of accessed attributes increases. This is because more data records or more data dimensions lead to higher volume of memory consumed. Figure 6.6(b) shows the memory usage when varying the number of generalization levels. From the graph, the more levels are included in each generalization hierarchy, the more memory is needed to store them, since the more levels there are, the more fine-grained the information becomes on each level, which enlarges the total memory usage.



(a) (b)

Figure 6.6: Time and space complexity when the number of generalization levels varies; (a) The running time when the number of generalization levels varies; (b) Peak memory when the number of generalization levels varies.

Effectiveness: Having verified the efficiency of our technique, we proceed to test its effectiveness. In this set of experiments, we use the *disclosure rate* to measure the effectiveness of our proposed access control model with generalization boundaries. We are going to use H, M and L to denote High, Medium and Low level in the classification of the generalization boundaries. Recall our privacy-aware access control model, if a data requester provides an access request, the access to each attribute is specified with generalization boundaries. Suppose there are n attributes in the database, among which there are n_H attributes which are generalized to $High$ level, n_M attributes are generalized to $Medium$ level, and n_L attributes are generalized to Low level, where $n_H + n_M + n_L = n$. In this case, the requester could totally access information in $n_H + 2n_M + 3n_L$ levels, which indicates the number of secure accesses. Consider the situation where there is no specification of generalization boundaries, for each attribute, the data requester could access any three-level information. Then there would be $3(n_H + n_M + n_L)$ accesses, and among those, there will be $3(n_H + n_M + n_L) - (n_H + 2n_M + 3n_L)$ insecure accesses. Thus, we define the *disclosure*



(a) (b) (c)

Figure 6.7: Disclosure rate comparison when varying (a) the number of H levels; (b) the number of M levels; (c) the number of L levels.

rate as $1 - \frac{n_H+2n_M+3n_L}{3(n_H+n_M+n_L)}$. The lower the rate is, the more secure the access control model would be.

The results are shown in Figure 6.7. Figure 6.7(a) displays the disclosure rate by varying the portion of L from 10% to 90%. From the graph, we can see that the disclosure rate is decreasing as the amount of L increases. This is expected, since the more the L level is specified in the generalization boundary, the less insecure accesses there are and the lower the disclosure rate is. Figure 6.7(b) describes the disclosure rate by varying M from 10% to 90%. The graph shows that the disclosure rate almost remains unchanged with the increased portion of M. Figure 6.7(c) reports the effect of H on the disclosure rate. When varying the portion of H from 10% to 90%, the disclosure rate is ascending. This indicates that the more H level attributes are specified in the generalization boundary, the more information would be disclosed in a traditional access control model, which demonstrate our proposed access model could better avoid information disclosure by specifying generalization boundaries. Therefore, in this case, our privacy-aware access model is superior to the traditional delegation model.

6.7 RELATED WORK

To date, several approaches have been reported that deal with various aspects of the problem of high-assurance privacy systems.

The W3Cs Platform for Privacy Preference (P3P) [120] allows web sites to encode their privacy practice, such as what information is collected, who can access the data for what purposes, and how long the data will be stored by the sites, in a machine-readable format. P3P enabled browsers can read this privacy policy automatically and compare it to the consumer's set of privacy preferences that are specified in a privacy preference language such as a P3P preference exchange language (APPEL) [121], also designed by the W3C. Even though P3P provides a standard means for enterprises to make privacy promises to their users, P3P does not provide any mechanism to ensure that these promises are consistent with the internal data processing. By contrast, the work in our paper not only provides an effective generalization strategy to maximize data privacy and usability, but also provides details on how to manage the valid access process. In particular, we propose a privacy-aware access control model based on the generalization techniques.

The concept of Hippocratic databases that incorporates privacy protection within relational database systems was introduced by Agrawal et al. [3]. The proposed architecture uses privacy metadata, which consists of privacy policies and privacy authorizations stored in two tables. Byun et al. [19, 18] presented a comprehensive approach for privacy preserving access control based on the notion of purpose. In the model, purpose information associated with a given data element specifies the intended use of the data element, and the model allows multiple purposes to be associated with each data element. The granularity of data labeling is discussed in detail in [19], and a systematic approach to implement the notion of access purposes, using roles and role-attributes is presented in [18]. Although these models do protect the privacy of data providers, they are rigid and do not provide ways to maximize

the utilization of private information. More specifically, in these models, the access decision is always binary; i.e., a data access is either allowed or denied as in most conventional access control models. Different from previous models, the novelty of our approach is that our model can provide a much finer level of access control as the access decision is based on the question of "how much information can be allowed for a certain user", rather than "is information allowed for a certain user or not". In other words, every piece of information is classified into different generalization levels and every user is assigned an authorization to access the private information.

Previous work on multilevel secure relational databases [50, 85] also provides many valuable insights for designing a fine-grained secure data model. In a multilevel relational database system, every piece of information is classified into a security level, and every user is assigned a security clearance. Based on this access class, the system ensures that each user gains access to only the data for which s/he has proper clearance, according to the basic restrictions. Byun and Bertino [20] proposed a new class of access control systems based on the notion of $micro\text{-}view$, which applied the idea of views at the level of the atomic components of tuples to an attribute value. However, the model in [20] is not a complete solution but rather it is aimed to show some of the capabilities. Some technical challenges raised by their model have been solved in our paper. One of the challenges is to design metrics for data privacy and data usability. We solve this challenge by introducing the privacy-aware generalization boundary technique, which can maximize the privacy and utility for both data providers and data users. Another challenge is concerned with the applicability to general-purpose access control, which we solve by providing a complete access control model with the implementation of access control policy. We also discuss the state transition and architecture of our privacy-aware access control model.

6.8 SUMMARY

In this chapter, we have considered a generalization boundary technique that can satisfy the requirements of both data providers and data users. Both privacy and usability of data items can be achieved when data items are generalized using this technique. Moreover, we present a privacy-aware access control model, where the trust-based decision policy and ongoing access control policy combine together to create a secure protection system. Further, our model provides a much finer level of control as the access control decision is based on the question of "how much information can be allowed for a certain user", rather than "is information allowed for a certain user or not". The privacy-aware access control model presented in this section provides an example of multi-level secure relational databases.

CHAPTER 7

OPTIMAL PRIVACY-AWARE PATH IN HIPPOCRATIC DATABASES

In this chapter, we present an approach to automatically derive the optimal way of authorizations needed to achieve a service from enterprise privacy policies. In particular, we organize purposes into purpose directed graphs through AND/OR decompositions, which support the delegation of tasks and authorizations when a host of partners participating in the business service provides different ways to achieve the same service. Further, we allow customers to express their trust preferences associated with each partner of the business process. Thus, a weight combining privacy cost and customer trust is given on each arc of the graph in the form of privacy penalties, and the process for fulfilling a purpose can be customized at runtime and guarantees minimal privacy cost and maximal customer trust because it was selected with the criterion of the optimal privacy penalty. Finally, an efficient algorithm is proposed to find the optimal privacy-aware path in Hippocratic databases. Our work is grounded on the modeling and analysis of purposes for Hippocratic databases and proposes enhancements to Hippocratic database systems in order to deal with inter-organizational business processes.

The information in this chapter is based on a published paper [63].

7.1 INTRODUCTION

With the widespread use of information technology in all walks of life, personal information is being collected, stored and used in various information systems. Achieving privacy preservation has became a major concern. Issues related to privacy have been widely investigated

and several privacy protecting techniques have been developed. To our best knowledge, the most well known effort is the W3C's Platform for Privacy Preference (P3P) [29]. P3P allows websites to express their privacy policy in a machine readable format so that using a software agent, consumers can easily compare the published privacy policies against their privacy preferences. While P3P provides a mechanism for ensuring that users can be informed about privacy policies before they release personal information, some other approaches are proposed [9, 18, 54, 69, 112], where the notion of $purpose$ plays an important role in order to capture the intended usage of information.

As enterprises collect and maintain increasing amounts of personal data, not only individuals are exposed to greater risks of privacy breaches and identity theft, but many enterprises and organizations are deeply concerned about privacy issues as well. Many companies, such as IBM and the Royal Bank Financial Group, use privacy as a brand differentiator [5]. By demonstrating good privacy practices, many business try to build solid trust with customers, thereby attracting more customers [89, 13, 15, 39]. Together with the notion of $purpose$, current privacy legislation also defines the privacy principles that an information system has to meet in order to guarantee a customer's privacy [38, 3, 4, 96]. A mechanism for negotiation is presented by Tumer et al. [96]. Enterprises specify which information is mandatory for achieving a service and which is optional, while customers specify the type of access for each part of their personal information.

On the basis of the solution for the exchange between enterprises and customers, Hippocratic databases enforced fine-grained disclosure policies to an architecture at the data level [3]. In the proposed architecture, enterprises declared the purpose for which the data are collected, who can receive them, the length of time the data can be retained, and the authorized users who can access them. Hippocratic databases also created a privacy authorization table shared by all customers, but it does not allow to distinguish which particular method is

used for fulfilling a service. Moreover, enterprises are able to provide their services in different ways, and each different method may require different data. For example, notification can be done by email or by mobile phone or by fax. Depending on the different kinds of methods, customers should provide different personal information. Asking for all personal information for different service methods as compulsory would clearly violate the principle of minimal disclosure.

On the server side, a single enterprise usually could not complete all the procedures of a service by itself, rather a set of collaborating organizations must participate in the service. Enterprises might need to decompose a generic purpose into more specific sub-purposes since they are not completely able to fulfill it by themselves, and so they may delegate the fulfillment of sub-purposes to third parties. It is up to customers to decide on a strategy of service fulfillment on the basis of their personal feeling of trust for different service components. A question that many customers have when interacting with a web server, with an application, or with an information source, is "Can I trust this entity?". Different customizations may require different data for which considerations may vary; there might be different trust levels for different partners (sub-contractors). The choice of service customization has significant impact on the privacy of individual customers.

The rest of this chapter is organized as follows. Section 7.2 introduces the motivation of this chapter based on a running example. Section 7.3 presents some background information on Hippocratic database systems.We introduce purpose directed graph with delegation in Section 7.4 and discuss how to characterize the privacy penalty and efficiently find the optimal solution in Section 7.5. We provide a brief survey of related work in Section 7.6. We summary this chapter in Section 7.7.

7.2 MOTIVATION

We consider the following example throughout the paper.

Example: *Ebay is an online seller in Australia and provides an online catalogue to its customers who can search for the items they wish to buy. Once customers have decided to buy goods, Ebay needs to obtain certain personal information from customers to perform purchase transactions. This information includes name, shipping address, and credit card number. Ebay views purchase, its ultimate purpose, as a three-step process: credit assessment, delivery, and notification. Credit assessment relies on Credit Card Company (CCC). Delivery can be done either by a delivery company or the post office, while notification can be done by email or by mobile phone.*

Obviously, Ebay provides many ways to achieve the purchase service and each different method could require different data. An important principle is that enterprises should disclose to customers which data is collected and for what purpose. Also, enterprises should maintain the minimal personal information necessary to fulfill the purpose for which the information was collected.

From the customers' point of view, they do not want to disclose more data than needed to get the desired service; rather, they want the process that best protects their privacy based on their preferences. Depending on the method of notification, Ebay needs either an email address or a mobile phone number. For example, Jimmy, a professor plagued by spam, may treasure his email address and give away his business mobile phone number. Bob, a doctor whose mobile phone is always ringing, may have the opposite preference. Therefore, it is up to customers to decide how to get a service fulfilled on the basis of their personal feeling of any service customization.

Furthermore, if we consider the delivery service, Ebay could not fulfill the service by

itself, but rather relies on a delivery company or post office. That means Ebay may outsource a large part of data processing to third parties participating in a single business process. However, the more the data is used, the more likely it might be disclosed, since the personal information is transmitted from one to another. This requires that enterprises maintain the minimal personal information necessary to fulfill the purpose. Moreover, the partners chosen by Ebay might also be trusted differently by its potential customers. The burden of choice is on the human who must decide what to do on the basis of his/her personal feeling of trust of the enterprises. For instance, Albert may prefer to delivery by a delivery company, since it is fast; whereas, Bob may chose delivery by post office because it is safe. Different partners (sub-contractors) chosen for the same purpose may have different trust levels. The choice of service customization has significant impact on the privacy of individual customers.

If we consider these factors, both the privacy cost and customer's trust should be considered as important factors in a privacy security system when enterprises publish comprehensive privacy policies involving hierarchies of purposes, possibly spanning multiple partners. Formally, it can be stated as follows:

Minimal privacy cost: Is there a way to fulfill the purpose with minimal privacy cost?

Maximal customer's trust: Is there a way to fulfill the purpose with maximal trust between enterprises and customers?

Classical privacy-aware database systems such as Hippocratic databases do not consider these issues. We are interested in solutions that support customers and companies alike, so that companies can publish comprehensive privacy policies involving multiple service methods, and possibly delegation of tasks and authorizations. Moreover, the solutions will allow customers to personalize services based on their own privacy sensitivities and their trust of partners who might contribute to the requested service.

7.3 OVERVIEW OF HIPPOCRATIC DATABASES

Hippocratic databases use $purpose$ as a central concept [3]. A purpose describes the reason(s) for data collection and data access, which is stored in the database as a "special" attribute occurring in every table of the database. This attribute specifies the purpose (reason/goal) for which a piece of information can be used.

For example, Table 7.1 shows the schema of two tables, customer and order, that store the personal information including purposes. In particular, table customer stores personal information about customers, and table order stores information about the transactions between enterprises and their customers. Then, for each purpose and data item stored in the database, we have:

External-recipients: the actors to whom the data item is disclosed;

Retention-period: the period during which the data item should be maintained;

Authorized-users: the users entitled to access the data item.

Purpose, external recipients, authorized users, and retention period are stored in the database with respect to the metadata schema defined in Table 7.2. Specifically, the above information is split into separate tables: external-recipients and retention period are in the $privacy\text{-}policies\ table$, while the authorized-users are in the $privacy\text{-}authorizations\ table$. The purpose is stored in both of them. The privacy-policies table contains the privacy policies of the enterprise, while privacy-authorizations table contains the access control policies that implement the privacy policy and represents the actual disclosure of information. In particular, privacy-authorizations tables are derived from privacy-policies tables by instantiating each external recipient with the corresponding users. Therefore, Hippocratic database systems define one privacy-authorizations table for each privacy-policies table, and these tables represent what information is actually disclosed.

table	attribute
customer	purpose, customer-id, name, address, email, fax-number, credit-card-info
order	purpose, customer-id, transaction, book-info, status

Table 7.1: Database schema

table	attribute
privacy-polices	purpose, table, attribute, {external-receipts}, {retention-period}
privacy-authorizations	purpose, table, attribute, {authorized-users}

Table 7.2: Privacy metadata schema

Hippocratic database system is an elegant and simple solution but does not allow for dynamic situations that could arise with web services and business process softwares. In such settings, enterprises may provide services in many different ways and may delegate the execution of parts of the service to third parties. This is indeed the case for a virtual organization based on a business process for web service where different partners explicitly integrate their efforts into one process [48].

7.4 PURPOSE DIRECTED GRAPH WITH DELEGATION

Agrawal et al. [3] proposed a structure to split a purpose into multiple purposes and then store them in the database. Karjoth et al. [54] used a directory-like notation to represent purpose hierarchies, which loses the logic relation between a purpose and its sub-purposes. In particular, this notation does not distinguish if a sub-purpose is derived by AND or OR decomposition [75]. Assuming a purpose p is AND-decomposed into sub-purposes $p_1, \ldots, p_n$, then all of the sub-purposes must be satisfied in order to satisfy p. For example, Ebay AND-decomposes purchase into delivery, credit assessment, and notification, then all of the three sub-purposes have to be fulfilled for fulfilling the purchase purpose. However, if a purpose p is OR-decomposed into sub-purposes $p_1, \ldots, p_n$, then only one of the sub-purposes must be satisfied in order to satisfy p. For instance, Ebay further OR-decomposes delivery into direct

delivery relying on delivery companies and delivery by post office. In this way, only one of them could be necessary to fulfill the delivery purpose. In essence, AND-decomposition is used to define the process for achieving a purpose, while OR-decomposition defines alternatives for achieving a purpose.

Our approach is based on traditional goal analysis [74], and consists of decomposing purposes into sub-purposes through an AND/OR refinement. The idea is to represent purpose hierarchies with directed graphs.

Definition 7.1. *A purpose directed graph PDG is a pair (P, A), where P is a set of purposes and A is the set of arcs, each arc represents a hierarchical relation between the purposes.*

A purpose directed graph (PDG) can be used to represent goal models in goal-oriented requirements engineering approaches [17]. For our purposes, they represent the entire set of alternative ways for delivering a service required by customers. Such representations can also be used to model the delegations of tasks and authorizations in the security modeling methodology proposed by Giorgini et al. [42].

An enterprise could provide different methods to achieve a service or rely on different partners to achieve the same part of the service. In particular, Ebay relies on a delivery company, Worldwide Express (WWEx), for shipping books. Ebay needs to delegate customer's information, such as name and shipping address, to WWEx. In turn, WWEx depends on local delivery companies for door-to-door delivery. To this end, WWEx delegates customer information to the local delivery companies LDC_1, $\cdots$, LDC_n for door-to-door delivery. Consequently, different processes can be used to fulfill the required service. To capture this insight, we introduce the notion of path.

Definition 7.2. *Let $PDG = (P, A)$ be a purpose directed graph. A path from v_0 to v_m is defined as a sequence $W = (v_0, a_1, v_1, ..., a_m, v_m)$, where a_i is an arc from v_{i-1} to v_i for $i = 1, ..., m$.*

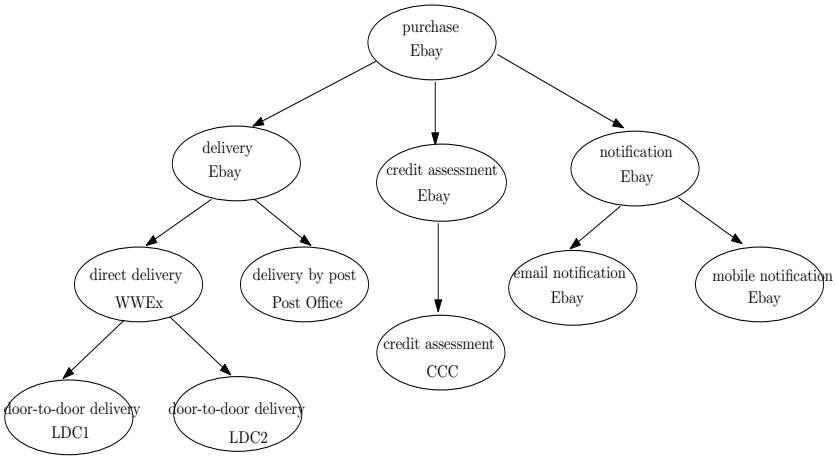


Figure 7.1: Purpose directed graph

A purpose directed graph PDG is rooted if it contains a vertex v, such that all the vertices of PDG are reachable from v through a directed path. The vertex v is called a root of PDG.

For example, consider the purpose directed graph depicted in Figure 7.1. Each vertex is composed of two parts: a purpose identifier and an enterprise needed to fulfill the purpose, and each of the purposes represents the policies of a single enterprise. The vertex 'purchase' is the root of the graph and purchase is the root-level purpose. Essentially, if a path $W = (v_0, a_1, v_1, ..., a_m, v_m)$ satisfies that v_0 is the root purpose and there exists no downward paths from v_m, we say the path is an essential path. An essential path represents a possible process through which an enterprise can fulfill the root purpose.

The enterprise-wide privacy policies are derived by looking at the Hippocratic database of each partner involved in the business process and merging them into a single purpose. Therefore, purposes can be recognized as the outcome of a process of refinements of goals in security requirements modeling methodologies [41]. The task delegation is indeed the case of a virtual organization based on a business process for web service where different partners explicitly integrate their efforts into one process.

7.5 FINDING OPTIMAL PRIVACY-AWARE PATH

Our goal is to decide which is the essential optimal privacy-aware path to fulfill the root purpose with respect to the customer's preference. This can be performed through the following quantitative analysis.

7.5.1 OBJECTIVE CHARACTERIZATION

Since our reference business model is that of virtual organizations, we assume that there will often be more than one way to deliver a service. Yet, they may differ in an important aspect, notably they may require different private data items, which incur different privacy costs. Further, depending on each customer's individual preferences, the same decomposition path might have significantly different trust values for different customers. In order to support quantitative analysis, we need to introduce the notion of privacy penalty.

Definition 7.3. *The privacy penalty of an arc a is defined as a pair $w_a = (\alpha, \beta)$, where α is the privacy cost and β is the customer's trust value on the arc a.*

Choice of α, β: The privacy penalty pair (α, β) on each arc can be pre-defined by asking the enterprises and customers to specify the level of privacy cost, and the trust they feel about the sub-suppliers. Since the personal information is transmitted from one to another, this may increase the danger of the leakage of personal information. Therefore, we use α to depict the privacy cost. Generally, we assume that there are different trust values based on the customer's personal feeling of the trust on different service customizations. For example, Bob prefers mobile notification more than email notification because of his personal experience, so there is a high trust value on mobile notification.

Intuitively, the privacy penalty of a path should consist of two parts: one is the sum of the privacy cost on each arc and the other is the minimum trust among these arcs.

Definition 7.4. *Let $\mathcal{P} = (v_0, a_1, v_1, \cdots, a_m, v_m)$ be a path in the PDG. Then, the privacy penalty of the path $\omega_{\mathcal{P}} = \omega_{a_1} + \ldots + \omega_{a_m} = (\sum_{i=1}^{m}(\alpha_i), min_{i=1}^{m}(\beta_i))$, where $\omega_{a_i} = (\alpha_i, \beta_i)$, $i = 1, ..., m$.*

Essentially, a path represents a possible process through which an enterprise can fulfill a root purpose. For our purpose, we use the sum of the private cost of each arc because we argue that the more a piece of data is used, the more likely it might be misused. The smaller the sum is, the less the privacy cost is. Therefore, sum measures are the ones that capture best one's intuitions on the cost of privacy. We also use the minimization function on trust values to get the smallest trust value on this path. The larger the value is, the more the trust is on this path. Our goal is to decide which is the process with the optimal privacy penalty (i.e., the minimal privacy cost and maximal trust value) to fulfill the root purpose with respect to the user's preferences. In order to describe the user's preference, we next introduce a flexible objective function.

Flexible objective function: If the privacy penalty on the arc a is defined as $w_a = (\alpha, \beta)$, we introduce the following objective function to balance the privacy cost and customer trust with a preference coefficient γ ($0 \leq \gamma \leq 1$).

$$alt(a) = \gamma \times \alpha + (1 - \gamma) \times \beta \tag{7.1}$$

The choice of parameter γ depends on the customer's preference. If the customer cares whether data are disclosed at all, then γ may be set with a value in the interval $0.5 \leq \gamma \leq 1$. On the other hand, if the customer stresses more on trust, then γ can be set with a value between 0 and 0.5.

In addition to the objective function, we propose to decompose purposes into sub-purposes through an AND/OR decomposition. In essence, AND-decomposition is used to define the

process for achieving a purpose, while OR-decomposition defines alternatives for achieving a purpose. Normally, the node purpose can be either AND-decomposed or OR-decomposed. A decomposition arc is either an OR-arc or an AND-arc.

Definition 7.5. *Let $PDG = (P, A)$ be a purpose directed graph, for each vertex $v \in P$, we denote $OUT(v) = OUT_{or}(v) \cup OUT_{and}(v)$ as the set of all successors of v, where $OUT_{or}(v)$ refers to all successors connecting v with OR-arcs, and $OUT_{and}(v)$ stores all successors connecting v with AND-arcs. Especially, if $OUT(v) = \emptyset$, we say the vertex v is a leaf of PDG.*

For example, in Figure 7.2 the root purpose r is AND-decomposed into three sub-purposes: delivery, credit assessment and notification, then $OUT(r) = OUT_{and}(r) = \{$delivery, credit assessment and notification$\}$. Further, considering the node v with purpose 'mobile notification', since $OUT(v) = \emptyset$, then the node 'mobile notification' is a leaf of the purpose directed graph.

7.5.2 THE ALGORITHM

In this section, we present efficient algorithms to track the optimal path that the enterprises need to fulfill a purpose. Before introducing the algorithm, the following lemma states the optimal substructure property of the privacy-aware paths, which is the theoretical foundation for our core algorithms.

LEMMA 7.1: *Given a purpose directed graph $PDG = (P, A)$ with privacy penalty function $\omega : A \rightarrow R$, let $\mathcal{P} = (v_1, v_2, \cdots, v_k)$ be a path with optimal privacy penalty from vertex v_1 to vertex v_k, and for any i and j ($1 \leq i \leq j \leq k$), let $\mathcal{P}_{ij} = (v_i, v_{i+1}, \cdots, v_j)$ be the subpath of $\mathcal{P}$ from vertex v_i to v_j. Then, $\mathcal{P}_{ij}$ is a path with optimal privacy penalty from v_i to v_j.*

Algorithm 1: $optimal\_path(PDG, OR\_r)$

Input: a purpose directed graph PDG with OR-decompoded root r.
Output: The optimal path D

1. Contract each vertex v with all its successors in $OUT_{and}(v)$ to the compound vertex v_c
2. Transfer PDG into $\overline{PDG}$
3. $\overline{p} = optimal\_path(\overline{PDG})$
4. If $\overline{p}$ contains compound vertex(vertices),
5. expand the compound vertex(vertices) on $\overline{p}$ to p,
6. $D = p$
7. else
8. $D = \overline{p}$

PROOF: If we decompose path $\mathcal{P}$ into $v_1 \xrightarrow{\mathcal{P}_{1i}} v_i \xrightarrow{\mathcal{P}_{ij}} v_j \xrightarrow{\mathcal{P}_{jk}} v_k$, then we have that $\omega_{\mathcal{P}} = \omega_{\mathcal{P}_{1i}} + \omega_{\mathcal{P}_{ij}} + \omega_{\mathcal{P}_{jk}}$. Now, assume that there is another path $\mathcal{P}'_{ij}$ from v_i to v_j with a lower privacy penalty, i.e. $\omega_{\mathcal{P}'_{ij}} < \omega_{\mathcal{P}_{ij}}$. Then, $v_1 \xrightarrow{\mathcal{P}_{1i}} v_i \xrightarrow{\mathcal{P}'_{ij}} v_j \xrightarrow{\mathcal{P}_{jk}} v_k$ is a path from v_1 to v_k whose privacy penalty is less than $\omega_{\mathcal{P}}$, which contradicts the assumption that $\mathcal{P}$ is the path with optimal privacy penalty from v_1 to v_k. ■

Next, we analyze two situations in finding the optimal privacy-aware path.

Case 1: if the root purpose is OR-decomposed, the algorithm consists of following steps:

- To contract each vertex v with all its successors in $OUT_{and}(v)$ to a compound vertex v_c; suppose $OUT_{and}(v) = \{v_1, \cdots, v_k\}$, we define $cost[v_c] = \sum_{i=1}^{k} \alpha(v, v_i)$, $trust[v_c] = \min_{i=1}^{k} \beta(v, v_i)$;

- To transfer the purpose directed graph PDG into $\overline{PDG}$ with no AND-arcs and find the optimal path $\overline{p}$ using function $optimal\_path(\overline{PDG})$;

- If the optimal path of $\overline{PDG}$ contains a compound vertex (or vertices), then expand the compound vertex (or vertices) on $\overline{p}$ to become the optimal solution of PDG.

function: $optimal\_path(\overline{PDG})$:

Input: $\overline{PDG}$ with root purpose r and leaves $v_1, \ldots, v_k$, pre-defined privacy cost and trust function $\alpha(*,*)$, $\beta(*,*)$, and preference coefficient $0 \leq \gamma \leq 1$

0 for each vertex v (not a compound vertex) in $\overline{PDG}$:
1 $cost[v] := 0$
2 $trust[v] := \infty$
3 for each leaf v_i $(i = 1, \cdots, k)$
4 $Sum(v_i) := 0$, $previous[v_i] := \{v_i\}$
5 while $r \notin predecessor[v_i] = \{u_{i_1}, \cdots, u_{i_s}\}$
6 {
7 for each u_{i_j} $(1 \leq j \leq s)$
8 $cost(u_{i_j}, v_i) := cost[v_i] + cost[u_{i_j}] + \alpha(u_{i_j}, v_i)$
9 $trust(u_{i_j}, v_i) := min\{trust[v_i], trust[u_{i_j}], \beta(u_{i_j}, v_i)\}$
10 $alt(u_{i_j}, v_i) := \gamma \times cost(u_{i_j}, v_i) + (1 - \gamma) \times trust(u_{i_j}, v_i)$
11 if $\gamma \geq 0.5$ /* prefer cost */
12 let $alt(u_{i_m}, v_i) = \min_{j=1}^{s} alt(u_{i_j}, v_i)$
13 $previous[v_i] := previous[v_i] \cup \{u_{i_m}\}$
14 $Sum(v_i) := Sum(v_i) + alt(u_{i_m}, v_i)$
15 $v_i := u_{m_i}$
16 if $\gamma < 0.5$ /* prefer trust */
17 let $alt(u_{i_m}, v_i) = \max_{j=1}^{s} alt(u_{i_j}, v_i)$
28 $previous[v_i] := previous[v_i] \cup \{u_{i_m}\}$
29 $Sum(v_i) := Sum(v_i) + alt(u_{i_m}, v_i)$
20 $v_i := u_{m_i}$
21 }
22 /*end while and all paths from the leaf to the root are found*/
23 if $\gamma \geq 0.5$
24 assume $Sum(v_t) = min_{i=1}^{k} Sum(v_i), (1 \leq t \leq k)$
25 output $previous[v_t]$
26 if $\gamma < 0.5$
27 assume $Sum(v_t) = max_{i=1}^{k} Sum(v_i), (1 \leq t \leq k)$
28 output $previous[v_t]$
29 end function

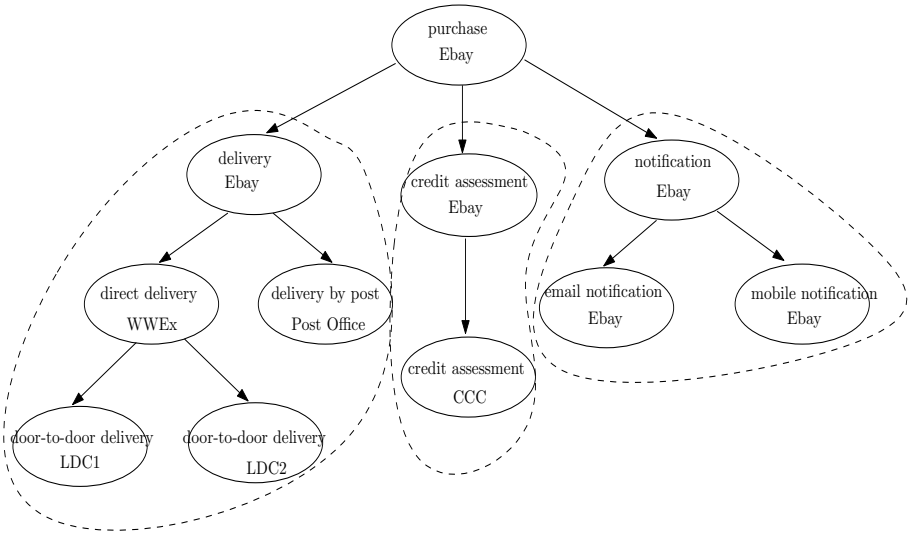


Figure 7.2: sub_PDG in Purpose directed graph

In function $optimal\_path(\overline{PDG})$, $\alpha(u, v)$ represents the privacy cost between the two nodes u and v, and $\beta(u, v)$ refers to the trust value on the arc (u, v). For each leaf vertex, Sum function is used to track the distance between the leaf and the root, while $predecessor[]$ records all predecessor vertices of the leaf, and $previous[]$ records the vertices on the optimal path from the leaf to the root. alt on line 10 is the objective function with the preference coefficient γ. If $\gamma \geq 0.5$, it means customers prefer more on privacy protection, in which case the minimal objective value is needed depending on the minimization function; while if $\gamma < 0.5$, it means customers prefer more on trust, then the maximal objective value is needed depending on the maximization procedure.

Case 2: if the root purpose is AND-decomposed, in order to design efficient algorithms to determine the process by which a service can be delivered with optimal privacy penalties, we need the definition of a sub-purpose directed graph.

Definition 7.6. *Let $PDG = (P, A)$ be a purpose directed graph, if the root purpose r is AND-decomposed into several sub-purposes, then each sub-purpose with all its descendants form a sub-purpose directed graph of PDG, and we denote it by sub_PDG. Essentially, if*

Algorithm 2: $optimal\_path(PDG, AND\_r)$

Input: A purpose directed graph PDG with AND-decomposed root
Output: The optimal path D

1. decompose PDG into several sub_PDG
2. for each sub_PDG with root r
3. if the root r is OR-decomposed in sub_PDG
4. run algorithm $optimal\_path(sub\_PDG, OR\_r)$
5. output $p_{or} = optimal\_path(sub\_PDG)$
6. if the root r is AND-decomposed in sub_PDG
7. further decompose the sub_PDG into several $sub_(sub_PDG)$s
2. for each $sub\_(sub\_PDG)$ with root r'
8. run algorithm $optimal\_path(sub\_(sub\_PDG), OR\_r')$
9. output $p_{and} = optimal\_path(sub\_PDG)$
10. $D = (\cup p_{or}) \cup (\cup p_{and})$

the root of the sub_PDG is further AND-decomposed into several sub-purposes, then each sub-purpose with all its descendants form a sub-purpose directed graph of sub_PDG, which is also a sub-sub-purpose directed graph of PDG, and we denote it by $sub\_(sub\_PDG)$.

For example, in Figure 7.2 Ebay AND-decomposes purpose purchase into three sub-purposes: delivery, credit assessment and notification. According to the definition of the sub-purpose directed graph, the purpose delivery with all its decedents consists of a sub-purpose directed graph. The same situation applies to the other two sub-purposes, so there are three sub-purpose directed graphs as in Figure 7.2 (circled in broken line). Since in each sub-purpose directed graph, the root is further OR-decomposed, there is no sub-sub-purpose directed graph in Figure 7.2.

For the sake of simplicity, we assume that the root of each $sub\_(sub\_PDG)$ is OR-decomposed. In this case, the algorithm consists of following steps:

- To decompose the purpose directed graph PDG into several sub-purpose directed graphs.
- For each sub-purpose directed graph sub_PDG with root purpose r,

- if the root purpose r is OR-decomposed, run algorithm $optimal\_path(sub\_PDG, OR\_r)$ to find the optimal path in sub_PDG;
- if the root purpose r is AND-decomposed, further decompose the sub_PDG into several sub-sub-purpose directed graphs, then run algorithm $optimal\_path$ $(sub\_(sub\_PDG), OR\_r')$ to find the optimal path in each $sub\_(sub\_PDG)$ with root r'. Combine all the optimal paths of each $sub\_(sub\_PDG)$ into the optimal solution of sub_PDG.

- To combine all the optimal paths of each sub_PDG into the optimal solution of PDG.

In Algorithm 2, p_{or} refers to the optimal path of sub_PDG with an OR-decomposed root, while p_{and} refers to the optimal path of sub_PDG with an AND-decomposed root.

Algorithm complexity: The time complexity of the two algorithms occurs mainly in the operation of function $optimal\_path()$, and the main computation cost of the function $optimal\_path()$ is spent on finding the vertices of the optimal privacy-aware path from each leaf vertex to the root purpose in PDG. If we take advantage of the vertices being numbered from 1 to $|P|$, for each leaf vertex, the effort for finding such an optimal path can be evaluated as $|P| + |P-1| + \cdots + 1 = \frac{|P|(|P|+1)}{2}$, and there are at most $|P| - 1$ leaf vertices, which makes the total execution time of $\frac{|P|(|P|+1)(|P|-1)}{2}$. So the time complexity for both algorithms are in $O(|P|^3)$.

Example: *Albert wants to buy some books. He prefers to receive books by delivery companies because it is fast. So, he defines his trust to WWEx higher than to the Post Office. Further, he prefers mobile notification more than email notification because of his personal experience. Thus, there is a higher trust value on mobile notification. The trust value described in Table 7.3 summarizes Albert's preferences and the privacy cost represents the online seller Ebay's default value to initiate the business process. Considering the privacy*

Service supplier	Privacy cost	Customer's trust
WWEx	0.2	0.7
LDC_1	0.4	0.5
LCD_2	0.2	0.3
Post Office	0.1	0.5
CCC	0.1	0.8
Email	0.1	0.4
Mobile	0.2	0.3

Table 7.3: Albert's personal preferences

cost defined by Ebay, the preference coefficient γ is set to 0.6, which means Albert places more importance on the privacy cost.

According to our algorithms, the first step is to AND-decompose the purchase purpose into sub-purposes: delivery, credit assessment and notification. Second, we need to find the optimal path in each sub-purpose directed graph. For example, in the sub-purpose directed graph with the root delivery purpose, there are three ways to fulfill the root purpose. Since Albert's preference coefficient value is $0.6 > 0.5$, then the minimal objective value is needed. Based on the different privacy cost and trust value on each path, the optimal path is "delivery $\rightarrow$ delivery by post". The same method can be applied to find the optimal paths in other sub-purpose directed graphs derived by credit assessment and notification purposes. Finally, combine all the optimal paths of each sub-purpose directed graph into the optimal solution for the purchase purpose highlighted in Figure 7.3 and it shows efficient balance between the privacy cost and the customer's trust.

7.6 RELATED WORK

Our work is related to several topics in the area of privacy and security for data management, namely privacy policy specification, privacy-preserving data management systems and multilevel secure database systems. We now briefly survey the most relevant approaches in these

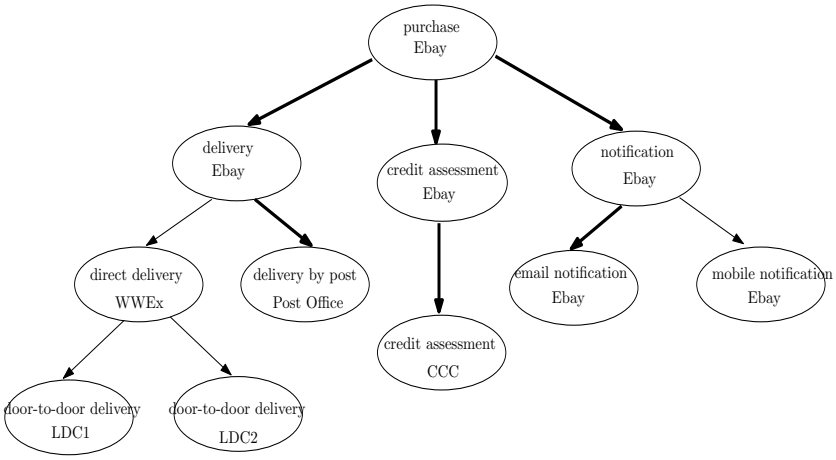


Figure 7.3: Optimal privacy-aware path

areas and point out the differences of our work with respect to these approaches.

The W3C Platform for Privacy Preference (P3P) [120] allows web sites to encode their privacy practice, such as what information is collected, who can access the data for what purposes, and how long the data will be stored by the sites, in a machine-readable format. P3P enabled browsers can read this privacy policy automatically and compare it to the consumer set of privacy preferences which are specified in a privacy preference language such as a P3P preference exchange language (APPEL) [29], also designed by the W3C. Even though P3P provides a standard means for enterprises to make privacy promises to their users, P3P does not provide any mechanism to ensure that these promises are consistent with the internal data processing. By contrast, the work in our paper provides an effective strategy to maximize privacy protection. Further, we allow customers to express their trust preferences associated with each partner of the business process in order to achieve maximal customer trust.

Byun et al. presented a comprehensive approach for privacy preserving access control based on the notion of purpose [19, 18]. In the model, purpose information associated with a given data element specifies the intended use of the data element, and the model allows multiple purposes to be associated with each data element. The granularity of data labeling

is discussed in detail in [19], and a systematic approach to implement the notion of access purposes, using roles and role-attributes is presented in [18]. Similar to our approach, they introduce purpose hierarchies in order to reason on access control. Their hierarchies are based on the principles of generalization and specification and are not expressive enough to support complex strategies defined by enterprises. However, we organize purposes into purpose directed graph through AND/OR decomposition, which supports the delegation of tasks and authorizations when a host of partners participating in the business process provides different ways to achieve the same service. We also present an efficient method to automatically derive the optimal way of authorizations needed to achieve a service from enterprise privacy policies.

The concept of Hippocratic databases, incorporating privacy protection within relational database systems, was introduced by Agrawal et al. [3]. The proposed architecture uses privacy metadata, which consist of privacy policies and privacy authorizations stored in two tables. LeFevre et al. [4] enhance Hippocratic databases with mechanisms for enforcing queries to respect privacy policies stated by an enterprise and customer preferences. In essence, they propose to enforce the minimal disclosure principle by providing mechanisms to data owners that control as who can access their personal data and for which purpose. Although the work on Hippocratic databases [3, 4] is closely related to ours, our approach has some notable differences. First, we introduce more sophisticated concepts of purpose, i.e., purposes are organized in a purpose directed graph through AND/OR decomposition. The second difference is that Hippocratic databases does not allow to distinguish which particular method is used; whereas, we discuss the situations that could arise with web services and business process software. Third, we provide an efficient method to automatically derive the optimal way of authorizations needed to achieve a service from enterprise privacy policies.

7.7 SUMMARY

In this chapter, we analyze the purposes behind the design of Hippocratic database systems, and organize them in hierarchal manner through AND/OR decomposition. We apply the purpose directed graph to characterize the ways the enterprise needs to achieve a service which may rely on many different partners. Specially, the selection of the partners and the identification of a particular plan to fulfill a purpose is driven by the customer's preferences. We use a goal-oriented approach to analyze the privacy policies of the enterprises involved in a business process, in which one can determine the minimum disclosure of data for fulfilling the root purpose with respect to the customer's maximum trust. On the basis of the purpose directed graph derived through a goal refinement process, we provide efficient algorithms to determine the optimal privacy-aware path for achieving a service. This allows the automatic derive action of access control policies for an inter-organizational business process from the collection of privacy policies associated with different participating enterprises.

CHAPTER 8

CONCLUSION

This chapter lists the contributions of this dissertation and provides directions for future work.

8.1 CONCLUSION

In this thesis we provided new authorization allocation algorithms for RBAC along with mobility based on relational algebra operations. In a system where there are hundreds of permissions and thousands of roles, it is very difficult to maintain consistency when using RBAC management. Especially, conflicts may arise when granting more than one permission as mobile or immobile members to a role in the permission-role assignment. We believe the approaches proposed in this thesis could automatically check the conflicts and help allocates permissions without compromising security. We also investigate how to revoke mobile or immobile permissions from a role. Even in the usage control model (UCON), less attention was put into discussing constraints associated with authorizations, obligations and conditions. Constraints in UCON are one of the most imponents that have been involved in the principle motivations of usage analysis and design. In this thesis, we provide a tool to precisely describe constraints and give out a formalized specification of usage control models. The flexibility and expressive capability of the specified UCON model are also show in this thesis.

In access control systems, managing users, permissions, roles and their interrelationships is a vital challenge. RBAC allows us to model security from the perspective of the delegation

of authority. The basic idea behind delegation is that some active entity in a system delegates authority to another active entity to carry out some functions on behalf of the former. Different delegation models are proposed, however, these models could not deal with the problem when delegating a collection of permissions to others. We proposed a flexible ability-based delegation model and analyzed the delegation framework, including delegating authorization and revocation with constraints. In an open environment, the entities are customarily alien to each other. When entering into a delegation, the delegator is entering into an uncertain interaction in which there is a risk of failure due to the delegation decisions. Therefore, the choice of the cooperative partner plays an important role in determining whether the delegation would success or not. We proposed a multi-level delegation model with trust management, where both delegation tasks and trust are organized into three levels. The delegation task levels are classified according to the information sensitivity, while, the trust levels combine reliability trust and future trust together to indicate to what extent a delegatee is reliable or trustworthy. The proposed multi-level delegation model allows a delegatee in a higher trust level to be assigned with a higher level of task. The experimental studies confirm the advantages of our model in terms of accurate prediction and sensitive information protection.

With the widespread use of information technology in all walks of life, privacy issues are exacerbated by the Internet. Traditional access control models can not help in privacy protection any more. We believe a new generation of access control models that could maximize data utility while minimizing disclosure of privacy is needed. In this thesis we proposed a privacy-aware access control model with generalization boundaries to balance privacy and information utility. We formalized authorizations with specific purposes and generalization levels and discussed how to manage a secure protection system in two stages. More specifically, a trust-based decision policy is proposed to handle access security with regard to the requester's trust at the first stage, and an ongoing access control policy is created to han-

dle the access security with regard to the retention period and generalization level at the second stage. Although Hippocratic databases enforced fine-grained disclosure policies by adopting the notion of purpose, the issues such as purpose hierarchies, task delegations and minimal privacy cost are missing from the proposed mechanism. In this thesis, we organized purposes into purpose directed graphs through AND/OR decomposition to characterize the ways the enterprise needs to achieve a service. We use a goal-oriented approach to analyze the privacy policies of the enterprises involved in a business process, in which one can determine the minimum disclosure of data for fulfilling the root purpose with respect to a customer's maximum trust. We further provide efficient algorithms to determine the optimal privacy-aware path for achieving a service, which allows to automatically derive access control policies for an inter-organizational business process from the collection of privacy policies associated with different participating enterprises.

8.2 FUTURE WORK

Based on the research work in this dissertation, we propose the following future research directions and issues:

- In this thesis we discussed the advanced permission-role relationship in RBAC and provided a formal language for specifying UCON models. Both access control and usage control contribute to security techniques for knowledge management. To build more efficient knowledge management system, further investigation into RBAC and UCON is needed as part of our future research work. We plan to provide secure strategies, processes, and metrics in our next research. Metrics must include support for security-related information. Processes must include secure operations. Strategies must include security strategies.

- In this thesis we extended the delegation model to satisfy ability-based delegation for groups of users. Moreover, we proposed multi-level delegation with trust management for multi-agent systems. This work opened up several directions for our future research. First, since delegation operations could temporarily change the access control state so as to allow an agent to use another agent's access privileges, colluding users may abuse the delegation support of access control systems to circumvent security policies, such as separation of duty. We intend to consider an enhanced form of delegation in order to avoid collusion in our future work. Second, the revocation of delegation has not discussed much in this thesis. It would be interesting to develop a revocation model to protect security under our multi-level delegation model.

- In this thesis we proposed a privacy aware access control model, which provided efficient generalization strategies for the preserving of privacy, but much more work still remains to be done. Future work includes devising a high level language in which privacy specifications can be expressed precisely. We also plan to extend our model to cope with complex query processing. We will introduce queries with join, sub-queries or aggregations into our model. These are challenging problems, but they are vital elements in a comprehensive privacy protection framework.

BIBLIOGRAPHY

[1] M. Abadi, M. Burrows, B. Lampson and G. Plotkin, A calculus for access control in distributed system, *ACM Trans. Program. Lang. Syst*. 15(4), 706-734, 1993.

[2] N. R. Adam and J.C. Worthmann, Security-control methods for statistical databases: a comparative study. CSUR, 21(4):515-556, 1989.

[3] R. Agrawal, J. Kiernan, R. Srikant and Y. Xu, Hippocratic Databases. *In: Proceedings of VLDB'02*, pp. 143-154. Morgan Kaufmann, San Francisco (2002)

[4] R. Agrawal, A. Evfimievski and R. Srikant, Information sharing across private databases. *In: Proceedings of SIGMOD'03*, pp. 86-97. ACM Press, New York (2003)

[5] P. Ashley, C.S. Powers and M. Schunter, Privacy promises, access control, and privacy management. *In: Third International Symposium on Electronic Commerce*, pp.13-16, (2002)

[6] V. Atluri and J. Warner, Supporting conditional delegation in secure workflow management systems. *In: SACMAT 2005: Proceedings of the tenth ACM symposium on Access control models and technologies*, pp. 49-58. ACM Press, New York (2005)

[7] E. Barka and R. Sandhu, Framework for Role-Based Delegation Models, *Proc of 16th Annual Computer Security Application Conference (ACSAC 2000)*.pp. 168-176, 2000.

[8] E. Barka and R. Sandhu, A Role-Based Delegation Model and Some Extensions, *Proc. of 23rd National Information Systems Security Conference (NISSC 2000)*. pp. 225-234, 2000.

[9] M. Backes, B. Pfitzmann and M. Schunter, A Toolkit For Managing Enterprise Privacy Policies. *In: Proceedings Of Esorics'03*, Lncs 2808, pp. 162-180. Springer, Berlin Heidelberg New York (2003)

[10] R. W. Baldwin, Naming and Grouping Privileges to Simplify Security Management in Large Database, Proceedings IEEE Computer Society Symposium on Research in Security and Privacy, pp. 184-194, 1990.

[11] D. E. Bell, and L. J. LaPadula, Secure Computer Systems: Mathematical Foundations and Model, Bedford, MA: The Mitre Corporation, 1973.

[12] D. F. C. Brewer, and M. J. Nash, The Chinese Wall Security Policy, Proceedings IEEE Computer Society Symposium on Research in Security and Privacy, April 1989, pp. 215-228.

[13] E. Bertino, E. Ferrari and A.C. Squicciarini, Trust-X: A Peer-to-Peer Framework for Trust Establishment. *IEEE Trans. Knowl. Data Eng.* 16(7): 827-842 (2004)

[14] E. Bertino, E. Ferrari and V. Atluri, Specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security*, 2(1), pp. 65-104, 1999.

[15] M. Blaze, J. Feigenbaum, and J. Lacy, Decentralized trust management, *In Proc. IEEE Symp. Security Privacy*, 1996, pp. 164-173.

[16] M. Blaze, J. Feigenbaum, J. Ioannidis and A. Keromytis. The role of trust management in distributed systm security, Security Internet Programming pp. 185-210.

[17] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini, TROPOS: An agent-oriented software development methodology. *JAAMAS* 8(3), 203-236 (2004)

[18] J. W. Byun, E. Bertino, and N. Li: Purpose based access control of complex data for privacy protection. *In: Proceedings of SACMAT'05*, pp. 102-110. ACM Press, New York (2005)

[19] J. W. Byun, E. Bertino and N. Li, Purpose based access control for privacy protection in relational database systems. *Technical Report* 2004-52, Purdue University.

[20] J. W. Byun, and E. Bertino, Micro-views, or on how to protect privacy while enhancing data usability: concepts and challenges. *SIGMOD Record* 35(1), pp. 9-13, (2006).

[21] P. Bonatti and P. Samarati, A Unified Framework for Regulating Access and Information Release on the Web. *In Journal of Computer Security*, 10, 3, (2002), pp. 241-271.

[22] D. E. Bell and L. J. Lapadula, Secure Computer Systems: Mathematical Foundations and Model. Mitre Corp. Report No.M74-244, 1975.

[23] E. Bertino, C. Bettini, and P. Samerati, A temporal authorization model. In Proceedings of ACM Conference on Computer and Communication Security. ACM, New York, pp. 126-135, 1994.

[24] E. Bertino, C. Bettini, and P. Samerati, A temporal access control mechanism for database systems. IEEE Transactions on Knowledge and Data Engineering 8, 1 (Feb.) pp. 67-80, 1996.

[25] E. Bertino, C. Bettini, and P. Samerati, An access control model supporting periodicity constraints and temporal reasoning. ACM Transaction on Database Systems 23, 3, pp. 231-285, 1999.

[26] C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekera, Obligation monitoring in policy management. In Proceedings of the 3rd InternationlWorkshop on Policies for Distributed Systems and Networks, pp. 2-35, 2002.

[27] C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekera, Provisions and obligations in policy management and security applications. In Proceedings of the 28th VLDB Conference, pp. 502-513, 2002.

[28] J. Chomicki, and J. Lobo, Monitors for history-based policies. In Proceedings of the 2nd Internationl Workshop on Policies for Distributed Systems and Networks, pp. 57-72, 2001.

[29] L. Cranor, M. Langheinrich, M. Marchiori, and J. Reagle: The platform for privacy preferences 1.0 (P3P1.0) specification. W3C recommendation (2002). http://www.w3.org/TR/P3P/

[30] J. Crampton and H. Khambhammettu, Delegation in role-based access control. *In: Proceedings of 11th European Symposium on Research in Computer Security*, pp. 174-191, (2006)

[31] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati and F. Violante. A reputation based approach for choosing reliable resources in peertopeer networks. *In Proceedings of ACM CCS02*, pp. 207-216, Washington DC, USA, November 2002.

[32] D. E. Denning, A lattice model of secure information flow, Communications of the ACM, vol. 19, no. 5, 1976.

[33] J. E. Dobson, and J. A. McDermid, Security Models and Enterprise Models, in Database Security, II: Status and Prospects, C. E. Landwehr, (ed.), New York: North Holland, 1989, pp. 1-39.

[34] H. L. Feinstein, R. Sandhu, C. E. Youman, and E. J. Coyne, Final Repon: Small Business Innovation Research (SBIR): Role-Based Access Control: Phase 1, McLean, VA, SETA Corporation, 24 January 1995.

[35] D. Ferraiolo, J. Barkley, and R. Kuhn, A role based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security*, 2(1), pp. 1-30, 1999.

[36] D. Ferraiolo and J. Barkley, Specifying and Managing Role-Based Access Control Within a Corporate Intranet. *Proc.of the 2ed ACM Workshop on Role-Based Access Control*. 1997, pp. 77-82.

[37] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn,and R. Chandramouli, Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224-274, 2001.

[38] E. Ferrari and B.M. Thuraisingham, Security and privacy for web databases and services. *In: Proceedings of the 9th International Conference on Extending Database Technology*, LNCS 2992, pp. 17-28. Springer, New York (2004).

[39] T. Finin and A. Joshi, Agents, trust, and information access on the semantic web, *ACM SIGMOD Record*, vol. 31, no. 4, pp. 30-35, Dec. 2002.

[40] Y. Frankel, Y. Tsiounis and M. Yung, Fair off-line e-cash made Easy. *in Advance in Cryptology, Proc. of Asiacrypt98*, LNCS 1294, PP. 257-270, 1998.

[41] P. Giorgini, F. Massacci, J. Mylopoulos and N. Zannone, Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning. *In Proc. of iTrust'04*, LNCS 2995, pp. 176-190. Springer-Verlag, 2004.

[42] P. Giorgini, F. Massacci, J. Mylopoulos and N. Zannone: Modeling security requirements through ownership, permission and delegation. *In: Proceedings of RE'05*, pp. 167-176. IEEE Press, Lausanne (2005)

[43] M. Gasser and E. McDermott, An Architecture for practical Delegation in a Distributed System, *IEEE Computer Society Symposium on Research in Security and Privacy*. pp. 20-30, 1990.

[44] C. Goh and A. Baldwin, Towards a more Complete Model of Role, *Proc. of 3rd ACM Workshop on Role-Based Access Control*. pp. 55-62, 1998.

[45] V.D. Gligor, S.T. Gavrila and D. Ferraiolo, On the formal denition of separation-of-duty policies and their composition. *in Proceedings of IEEE Symposium on Research in Security and Privacy*, pp. 172-183, Oakland, CA, May 1998.

[46] N. Griffiths, Task Delegation using Experience-Based Multi-Dimensional Trust, *In the Proceedings of the Fourth International Conference on Autonomous Agents and Multi-agent Systems (AAMAS-05)*, Utrecht, The Netherlands, 2005, pp. 489-496.

[47] T. Hardjono, T. Chikaraishi and T. Ohta, Secure Delegation of Tasks in Distributed Systems. *In Proceedings of the 10th International Symposium on the TRON Project*, pp. 98-112, Los Alamitos, California, USA, 1993.

[48] C. Handy: Trust and the virtual organization. Harv.Bus.Rev.73,40-50(1995)

[49] M. H. Harrison, W. L. Ruzzo, and J. D. Ullman, Protection in Operating Systems, Communication of ACM, Vol 19, No. 8, 1976.

[50] S. Jajodia, and R. Sandhu, Toward a multilevel secure relational data model. *In: ACM International Conference on Management of Data* (SIGMOD) pp. 50-59. ACM Press, New York (1991).

[51] J. Joshi, E. Bertino, B. Shafiq, and A. Ghafoor, Constraints: Dependencies and separation of duty constraints in gtrbac. In Proceedings of the 8th ACM Symposium on Access Control Models and Technologies, pp. 51-64, 2003.

[52] J. Joshi, E. Bertino, B. Shafiq, and A. Ghafoor, A generalized temporal role-based access control model. IEEE Transactions on Knowledge and Data Engineering 17, pp. 4-23, 2005.

[53] J. B. D. Joshi and E. Bertino, Fine-grained role-based delegation in presence of the hybrid role hierarchy. *In: SACMAT 2006: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pp. 81-90. ACM Press, New York (2006)

[54] G. Karjoth, M. Schunter and M. Waidner: Platform for enterprise privacy practices: privacy-enabled management of customer data. *In: Proceedings of PET'02*, LNCS 2482, pp. 69-84. Springer, Berlin Heidelberg New York (2002)

[55] S. D. Kamvar, M.T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. *In Proceedings of the 12th International WWW Conference*, pp. 640-651, Budapest, Hungary, May 2003.

[56] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. DeWitt, Limiting Disclosure in hippocratic databases. *In: The 30th International Conference on Very Large Databases* (VLDB), pp. 108-119, (2004)

[57] N. Li and B. N. Grasof, A practically implementation and tractable delegetion logic, IEEE Symposium on Security and Privacy, pp. 27-42.

[58] L. Li, Y. Wang and V. Varadharajan, Fuzzy Regression Based Trust Prediction in Service-Oriented Applications, *the Sixth International Conference on Autonomic and Trusted Computing (ATC-09)*, pp. 221-235, Brisbane, Australia, 7-9 July, 2009.

[59] M. Li, and H. Wang, ABDM: An Extended Flexible Delegation Model in RBAC, *the IEEE 8th International Conference on Computer and Information Technology (CIT'2008)*, pp.390-395, 2008, Sydney, Australia.

[60] M. Li, H. Wang and D. Ross. Trust-based Access Control for Privacy Protection in Collaborative Environment, *the 2009 IEEE International Conference on e-Business Engineering (ICEBE 2009)*, pp.425-430, Macau, China, October, 2009.

[61] M. Li, and H. Wang. Protecting information sharing in distributed collaborative environment. *In: 10th Asia-Pacific Web Conference Workshop (APWeb 2008)*, pp.192-200, 2008, Shenyang, China.

[62] M. Li, H. Wang, A. Plank, and J. Yong. Advanced permission-role relationship in role-based access control. *In: 13th Australasian Conference on Information Security and Privacy (ACISP 2008)*, pp.391-403, 2008, Wollongong, Australia.

[63] M. Li, X. Sun, H. Wang, and Y. Zhang. Optimal Privacy-aware Path in Hippocratic Databases. *to appear in 14th International Conference on Database Systems for Advanced Applications (DASFAA 2009)*, pp.441-455, Brisbane, Australia, April, 2009.

[64] M. Li, H. Wang, and A. Plank,. Privacy-aware Access Control with Generalization Boundaries. *to appear in 32nd Australasian Computer Science Conference (ACSC 2009)*, CRPIT 91, pp.93-100, 2009, Wellington, New Zealand.

[65] M. Li, and H. Wang. Specifying usage control model with object constraint language. *to appear in the 2010 International Conference on Data and Knowledge Engineering (ICDKE)*, Melbourne, Australia, 1-3 September 2010.

[66] M. Li, and H. Wang. Multi-level delegations with trust management in access control systems. *Submitted to Journal of Intelligent Information Systems*, 2010.

[67] C. Lin, and V. Varadharajan, Trust Enhanced Security for Mobile Agents. *in Proc of the 7th IEEE International Conference on E-Commerce Technology, CEC 2005, Germany*, July 2005, ISBN 0-7695-2277-7 ISSN 1530-1354.

[68] S. Marti and H. Garcia-Molina. Limited reputation sharing in P2P systems. *In Proceedings of ACM EC04*, pp. 91-101, New York, USA, May 2004.

[69] F. Massacci and N. Zannone: Privacy is linking permission to purpose. *In: Proceedings of the 12th International Workshop on Sec. protocols*, pp.192-198, (2004)

[70] S. Na, and S. Cheon, Role delegation in role-based access control. *In: RBAC 2000: Proceedings of the fifth ACM workshop on Role-based access control*, pp. 39-44. ACM Press, New York (2000)

[71] N. Nagaratnam and D. Lea, Secure Delegation for Distributed Object Environments, *USENIX Conference on Object Oriented Technologies and Systems*. pp. 8, 1998.

[72] M. Nash, and K. Poland, Some Conundrums Concerning Separation of Duty, IEEE Symposium on Security and Privacy, pp. 201-207, Oakland, CA, 1990.

[73] M. Nyanchama, and S. L. Osborn, Access Rights Administration in Role-Based Security Systems, Proceedings of the IFIP WG11.3 Working Conference on Database Security, pp. 37-56, 1994.

[74] N. J. Nilsson, Problem solving methods in AI. McGraw-Hill, 1971.

[75] N. J. Nilsson, Principles of Artificial Intelligence, Morgan Kaufman, 1994.

[76] M. Nyanchama and S. Osborn, The Role Graph Model and Conflict of Internet, *ACM Transaction on Information and System Security*, 2(1), 1999, pp. 3-33.

[77] T. J. Norman, and C. A. Reed, A Model of Delegation for Multi Agent Systems. *In Foundations and Applications of Multi Agent Systems*, volume 2403 of Lecture Notes in Artificial Intelligence, Springer-Verlag, pages 185-204, 2002.

[78] W. Nejdl, D. Olmedilla, and M. Winslett, PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. *In Proceedings of the Workshop on Secure Data Management in a Connected World (SDM 04) in conjunction with 30th International Conference on Very Large Databases*, pp. 118-132, 2004.

[79] T. Okamoto, An efficient divisible electronic cash scheme, *the 15th Annual International Cryptology Conference on Advances in Cryptology*, LNCS 963, Springer-Verlag, pp. 438-451,1995.

[80] F. Rabitti, E. Bertino, W. Kim, and D.Woelk. A model of authorization for next-generation database systems. In ACM Transactions on Database Systems (TODS), pp. 88-131, 1991.

[81] S. D. Ramchurn, C. Sierra, L. Godo, and N. R. Jennings. A computational trust model for multi-agent interactions based on confidence and reputation. *In Proc. of the 6th Int. Workshop of Deception*, Fraud and Trust in Agent Societies, pp. 69-75, 2003.

[82] M. Richters and M. Gogolla: On Formalizing the UML Object Constraint Language OCL. In Tok-Wang Ling etc editor: 17th International Conference on Conceptual Modeling (ER). Vol. 1507 Springer-Verlag (1998) 449-464.

[83] R. Sandhu and J. Park, Usage Control: A Vision for Next Generation Access Control, The Second International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security, 2003.

[84] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman, Role-Based Access Control Models, IEEE Computer, Volume 29, Number 2, February 1996.

[85] R. Sandhu and F. Chen, The multilevel relational data model. *ACM Trans. Inf. Syst. Secu.* 1(1), pp.93-132 (1998).

[86] R. Sandhu, Role Hierarchies and Constraints for Lattice-Based Access Controls. *In: European Symposium on Research in Security and Privacy* pp. 65-79, (1996).

[87] R. Sandhu, V. Bhamidipati and Q. Munawer, The ARBAC97 model for role-based administration of roles, *ACM Transaction on Information and System Security*, 1(2), 1999, pp. 105-135.

[88] R. Sandhu and Q. Munawer, The ARBAC99 Model for Administration of Roles, *in the Annual Computer Security Applications Conference*, ACM Press, pp. 229-238, 1999.

[89] K.E. Seamons, M. Winslett, T. Yu, L.Yu and R. Jarvis: Protecting privacy during on-line trust negotiation. *In: Proceedings of PET'02*, LNCS 2482, pp. 129-143. Springer, Berlin Heidelberg New York (2002)

[90] K. Seamons,M. Winslett, and T. Yu, Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. *In Proc. of NDSS*, pp. 109-125. IEEE Press, 2001.

[91] L.A. Stein, Delegation Is Inheritance, *Proc. of Object-Oriented Programming Systems, Languages, and Applications (OOPSLA'87)*. pp. 138-146, 1987.

[92] X. Sun, H. Wang, J. Li, and T.M. Truta, Enhanced P-Sensitive K-Anonymity Models for Privacy Preserving Data Publishing. *Transactions on Data Privacy* 1(2): 53-66 (2008)

[93] X. Sun, H. Wang, and J. Li, L-Diversity Based Dynamic Update for Large Time-Evolving Microdata. *Australasian Conference on Artificial Intelligence* pp. 461-469, 2008.

[94] L. Sweeney, Achieving k-Anonymity Privacy Protection Using Generalization and Suppression, *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, Vol. 10, No. 5 (2002), pp. 571-588.

[95] D. J. Thomsen, Role-Based Application Design and Enforcement, in Database Security, IV: Status and Prospects, S. Jajodia and C. E. Landwehr, (eds.), New York: North Holland, 1991, pp. 151-168.

[96] A. Tumer, A. Dogac and H. Toroslu, A Semantic based Privacy framework forweb services. *In: Proceedings of ESSW'03* , pp. 289-305, (2003).

[97] I.B. Vapnyarskii, "Lagrange multipliers", *In Hazewinkel, Michiel, Encyclopaedia of Mathematics*, Kluwer Academic Publishers, ISBN 978-1556080104.

[98] J. Wainer and A. Kumar, A fine-grained, controllable, user-to-user delegation method in rbac. *In: SACMAT 2005: Proceedings of the tenth ACM symposium on Access control models and technologies*, pp. 59-66. ACM Press, New York (2005)

[99] S. Waner and S.R. Costenoble. Applied Calculus. 4th edition. Apr 2007. Publisher: Brooks/Cole Pub Co.

[100] H. Wang, J. Li, R. Addie, S. Dekeyser and R. Watson, A framework for Role-based group delegation in distributed environment, *the 29th Australasian Computer Science Conference (ACSC2006)*, pp. 321-328, Australian Computer Society, Hobart, Australia, 2006.

[101] H. Wang and J. Cao, Delegating revocations and authorizations, accepted by the 1st International Workshop on Collaborative Business Processes, pp. 293-305, Brisbane, Australia, 2007

[102] H. Wang, J. Cao and Y. Kambayashi, Building a Consumer Anonymity Scalable Payment Protocol for the Internet Purchases, *The 12th International Workshop on Research Issues on Data Engineering: Engineering E-Commerce/E-Business Systems*, pp. 159-168, 2002, San Jose, USA.

[103] H. Wang, J. Cao and Y. Zhang, Formal authorization approaches for permission-role assignment using relational algebra operations, *Proceedings of the 14th Australasian Database Conference*, Feb. 2-7, 2003, Adelaide, Australia, Vol. 25, No.1, pp. 125-134.

[104] H. Wang, J. Cao and Y. Zhang, Formal Authorization Allocation Approaches for Role-Based Access Control Based on Relational Algebra Operations, *The 3nd International Conference on Web Information Systems Engineering (WISE'2002)*, pp. 301-310. Dec. 3-6, 2002, Singapore.

[105] Y. Wang and V. Varadharajan. Interaction trust evaluation in decentralized environments. *In Proceedings of 5th International Conference on Electronic Commerce and Web Technologies (EC-Web04)*, volume LNCS 3182, Springer-Verlag, pp. 144-153, Zaragoza, Spain, August-September 2004.

[106] Y. Wang, and J. Vassileva, Trust and reputation model in collaborative networks. *in Proc. 3rd IEEE Int. Conf. Collaborative Computing*, pp.150- 157, 2003.

[107] A. Westin, E-commerce and privacy: What net users want. *Technical report*, Louis Harris and Associates, June 1998.

[108] A. Westin, Freebies and privacy: What net users think. *Technical report*, Opinion Research Corporation, July 1999.

[109] W. Winsborough, and N. Li, Towards Practical Automated Trust Negotiation. *In Third International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*, Monterey, CA, June 2002.

[110] Z. Xie and C.H. Chi: Quantifying Trust through Delegation in Service Oriented Architecture. *IEEE SCW 2007*, pp. 308-315.

[111] L. Xiong and L. Liu. PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. on Knowlege and Data Engineering*, 16(7), pp. 843-857, 2004.

[112] M. Yasuda, T. Tachikawa and M.Takizawa, Information flow in a purpose-oriented access control model. *In: Proceedings of ICPADS'97*, pp. 244-249. IEEE Press, Lausanne (1997)

[113] G. Zacharia and P. Maes. Trust management through reputation mechanisms. *Applied Artificial Intelligence Journal*, 9:881-908, 2000.

[114] L. Zhang, G.J. Ahn, and B.T. Chu, A rule-based framework for role-based delegation and revocation. *ACM Trans. Inf. Syst. Secur.* 6(3), pp. 404-441 (2003)

[115] X. Zhang, S. Oh, and R. Sandhu, Pbdm: a flexible delegation model in rbac. *In: SACMAT 2003: Proceedings of the eighth ACM symposium on Access control models and technologies*, pp. 149-157. ACM Press, New York (2003)

[116] X. Zhang, J. Park, F. Parisi-Presicce, and R. Sandhu, A Logical Specification for Usage Control, In Proc. of the 9th ACM Symposium on Access Control Models and Technologies, pp. 1-10, 2004.

[117] L. Zhang, G.J. Ahn, and B.T. Chu, A rule-based Framework for Role-Based Delegation, *Proc. 6th ACM Symposium on Access Control Models and Technologies (SACMAT 2001)*, pp.153-162, May, 2002.

[118] M. Zurko, R. Simon and T. Sanlippo, A user-centered modular authorization service built on an rbac foundation. *In Proceedings of IEEE Symposium on Research in Security and Privacy*, pp. 57-71, Oak-land, CA, May 1999.

[119] IBM. The Enterprise Privacy Authorization Language (EPAL). Available at www.zurich.ibm.com/security/enterprise-privacy/epal.

[120] World Wide Web Consortium (W3C). A P3P Preference Exchange Language 1.0 (APPEL 1.0). Available at www.w3.org/TR/P3P-preferences.

[121] World Wide Web Consortium (W3C). Platform for Privacy Preferences (P3P). Available at www.w3.org/P3P