A Thesis

Entitled

A Flexible Role-Based Delegation Model and Its Application in Healthcare Information System

By

Zidong Liu

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Engineering

Dr. Weiqing Sun, Committee Chair

Dr. Mansoor Alam, Committee Co-Chair

Dr. Hong Wang, Committee Member

Dr. Patricia R. Komuniecki, Dean
College of Graduate Studies

The University of Toledo

August 2013
An Abstract of

A Flexible Role-Based Delegation Model and Its Application in Healthcare Information System

By

Zidong Liu

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Masters of Science Degree in Engineering

The University of Toledo

August 2013

As information systems became widely used by organizations and enterprises, resource sharing and collaboration of work have been pervasive. As a natural way to realize this, delegation has become the routine rather than the exception. However, traditional delegation models have encountered various issues in meeting the growing and diverse requirements of different industries and customers. Some of them fail to provide sufficient delegation functionalities, while others are cumbersome to apply and manage practically. Therefore it is imperative to propose a flexible delegation model that provides fine-grained control in response to different scenarios. Meanwhile, such a model should be easy to apply and maintain. Base on that, this study set out to design a flexible delegation model based on role-based access control which supports fine-grained delegation control at both role and permission levels. Moreover, it introduces a dynamic delegation role structure to deal with different types of delegation requests so that the delegation model can cope with unexpected delegation cases. Finally, a prototype is implemented based on a healthcare information system to demonstrate the feasibility of the model.
Acknowledgements

First and foremost I would like to heartfeltly thank my advisor Dr. Sun. The work presented in this thesis is not possible without his support and advice. I also appreciate all his contributions of time, ideas, and funding to make my research inspirable and stimulating. The insight and enthusiasm he has for his research was quite motivational during the period in my academic pursuit.

I would also like to express my gratitude to my co-advisor, Dr. Mansoor Alam, for his guidances and advices throughout my graduate studies. Such guidances and advices not only encourage me in the research work, but also assist me a in my personal life.

I would like to thank Dr. Wang for agreeing to serve on my committee.
# Contents

Abstraction ................................................................. iii
Acknowledgements ......................................................... iv
Contents ................................................................. v
List of Tables ................................................................. vii
List of Figures ................................................................. vii
List of Abbreviations ....................................................... viii

## 1 Introduction

1.1 Access Control and Delegation ................................................................. 1
1.2 Delegation in the Healthcare Domain .......................................................... 5
1.3 Objectives of the Project ........................................................................... 7
1.4 Synopsis of Thesis ................................................................................... 8
1.5 Outline of Thesis .................................................................................... 9

## 2 Literature Review

2.1 Outline .................................................................................................. 10
2.2 The NIST RBAC Model and Delegation ................................................... 10
2.3 Delegation Models in RBAC .................................................................. 15
2.4 Healthcare Applications ................................................................. 18
2.5 Open Source Electronic Health Record Systems ......................... 21

3 Delegation Architecture ................................................................. 24
  3.1 Outline ....................................................................................... 24
  3.2 Dynamic Delegation Role Structure ......................................... 24
  3.3 Delegation Decision Function .................................................. 30
  3.4 Multi-step Delegation ............................................................... 32
  3.5 Delegation Revocation ............................................................. 34

4 Implementation and Evaluation ..................................................... 38
  4.1 Outline ....................................................................................... 38
  4.2 Implementation ......................................................................... 38
  4.3 Programming and Database .................................................... 45
  4.4 Evaluations ............................................................................... 49

5 Conclusion and Future Work ......................................................... 54
  5.1 Summary and Conclusions ...................................................... 54
  5.2 Future Work ............................................................................. 55

References ....................................................................................... 56
## List of Tables

3. 1 Categorization of permissions and characteristics ......................................................27
3. 2 Permissions of physician, assistant doctor and PDR .....................................................29
4. 1 User information ........................................................................................................46
4. 2 User information controlled by phpGACL .................................................................47
4. 3 Permission information controlled by phpGACL .........................................................48
4. 4 Role_Permission_Map .................................................................................................48
4. 5 Rates of fulfilled TDRs ..............................................................................................48
4. 6 Information of successful and failed delegation .........................................................49
4. 7 Performance evaluation ..............................................................................................51
# List of Figures

2- 1 The simplest version of RBAC ................................................................. 12
2- 2 A role hierarchy example in a hospital .................................................. 13
2- 3 Classical RBAC96 model ....................................................................... 14
2- 4 Basic delegation model ......................................................................... 14
3- 1 A flexible delegation model with role structure .................................. 25
3- 2 Example of PDR roles in the role hierarchy .......................................... 29
3- 3 Process flow-chart of the model ............................................................. 32
4- 1 The delegation system .......................................................................... 39
4- 2 Delegation request form ....................................................................... 41
4- 3 Delegatee selection ................................................................................ 42
4- 4 Selected permissions from OpenEMR system ....................................... 44
4- 5 PhpGACL component to control access rights ...................................... 45
List of Abbreviations

BTG ......................... Break the Glass
DAC ........................ Discretionary Access Control
GACL ...................... Generic Access Control
MAC ........................ Mandatory Access Control
NIST ........................ National Institute of Standards and Technology
ONC ........................ Office of the National Coordinator
PA ............................. Permission Assignment
PBDM ........................ Permission-Based Delegation Model
PDR ............................ Predefined Delegation Role
RBAC ........................ Role Based Access Control
RBDM ........................ Role Based Delegation Mode
RDR ............................ Retained Delegation Role
RH ............................... Role Hierarchy
TDR ............................ Temporary Delegation Role
UA ............................... User Assignment
Chapter 1

Introduction

1.1 Access Control and Delegation

No matter in which domain, commercial, medical or educational, access control is an important security issue. Access is the ability to enter into a computer resource, and access control is a means by which the ability to access is explicitly enabled or restricted in some way [1]. Normally access control could be classified into three categories [2] - discretionary access control (DAC), mandatory access control (MAC) and role-based access control (RBAC). In RBAC, individual users are grouped into roles that are related to their positions within an organization and assigned permissions to various roles according to their statures in the organization [3]. Meanwhile, when compared to MAC and DAC, RBAC is considered to be more flexible and easier to manage in large scale organizations. Hence, up till now, RBAC has become dominant in the access control domain and has been widely used by the majority of organizations, especially large firms
enterprises, hospitals, and colleges due to its simplicity in the management of access
rights [4].

In current RBAC system, security officers handle assignments of users to roles. However, fully depending on this functionality may increase management efforts in a distributed environment because of the continuous involvement from security officers [5]. In this case, delegation would be a necessary approach to enhance the scalability because it enables decentralization of administration task. Delegation requirements arise when a user need to act on another’s behalf for accessing resources. In general, the basic idea of 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 [6] [7] [8].

Delegation may occur in two forms: administrative delegation and user delegation. An administrative delegation allows an administrative user to assign access rights to a user and does not, necessarily, require that the administrative user possesses the ability to use the access right. A user delegation allows a user to assign a subset of his/her available rights to another user. However, a user delegation operation requires that the user performing the delegation must possess the ability to use the access right [9]. Furthermore, it is believed that an administrative delegation operation is often long-lived and more durable (permanent) than a user delegation operation that is short-lived.
(temporary) and intended for a specific purpose [10]. Rather than normal access right administration operations, which are performed centrally, delegation operations are usually performed in a distributed manner [11].

Delegation can take place in many situations; the first one is backup of roles. Such a scenario is pervasive and takes place when an individual is on a business trip or long-term absence, the job functions need to be maintained by others. This requires that somebody else be delegated the authority to do the absent individual’s job. The second one is decentralization of authority, this happens when an organization needs to setup initially or reorganize subsequently; job functions are distributed from higher job positions to lower job positions in the organization structure. Delegation is also useful in collaboration of work, Oftentimes people need to collaborate with others in the same organization or other organizations. In this case, we need to grant some access authority to share information [12]. As administrative delegation is usually considered to be long-lived, the second case also needs durable delegation. On the other hand, the first and third cases need temporary delegation of authority.

Due to rapid growth of electronic commerce, the online information system has become the mainstream in large organizations. And all the resources required to carry out certain operations are barely local to the system where the user is logged in. In such cases,
information sharing tends to be very dynamic and in most times ad hoc; so delegation is more often the rule than the exception [13]. It is important to have a system that allows delegation, in order to simplify the administrator task and to manage collaborative work securely, in particular, with the increase of shared information and distributed systems [14]. Under these circumstances, the delegation model must be secure and flexible in order to guarantee that the security policy is not violated by delegation. Meanwhile, to cope with different delegation scenarios and provides multiple delegation features such as multi-step delegation and revocation. Moreover, it is also important to reduce administrative costs without compromising the capabilities of the access control system. However, traditional delegation models have encountered various issues in meeting the diverse and growing delegation requirements. For one thing, some existing approaches are focused on the theoretical models without the consideration of practical implementations and management costs. For another, although the traditional RBAC model can make authorization decision to enforce rules on a user’s role, it not sufficiently powerful to enforce delegation rules that are dependent on runtime parameters [15]. In other words, delegation should be adaptive and context-aware to environments and the delegation authentication and authorization should be enforced dynamically.
1.2 Delegation in the Healthcare Domain

Nowadays, many industries are extending their business operations to operate over the web, and healthcare is no exception [15]. However, unlike commercial access control system for large corporations and organizations, the health domain has its unique characteristics which strongly impact the needs for a coherent but flexible approach to manage delegation and access rights [16]. In healthcare systems, privacy is always been an important requirement. The access to patients’ medical records should be controlled strictly. However, in some emergency cases, there has to be a sophisticated mechanism to enter into a certain patient record. For instance, there might be emergency situations where the intern doctors need the access to the patient records. In this situation, an intern doctor would be able to give proper medication based on previous medical history of the patient. Otherwise, it may put the patient’s life at risk. So the intern doctor must have read access to check the history of the patient and write access to modify or append the drugs given at this moment.

Here is a case study by Carol J. Duh which gives the details of how important it is to have reasonable access to the patient information for an intern doctor in emergency situation. “A careful look at the Libby Zion case reveals that while her tragic death happened while intern was on 36-hour shift, the intern also did not have access to the list
of drugs that Libby was taking, and prescribed a drug that triggered an adverse interaction that ultimately led to her death. Timely and accurate information saves lives”.

For this the intern doctor reaction was “Although I was more tired during my rotation than I’ve been in my entire life, the complications I observed were less from exhaustion and far more from the challenge of handing off a patient from one clinician to another or organizing and accessing information”. She says that lab value or the patient’s vitals, is meaningless without the constellation of yesterday’s numbers and the clinical picture that surrounds it. This implies that, timely and easy access to patient information is crucial to a physician’s ability to provide proper care. Also, the intern doctor should have some practical experience in handing of a patient to another clinician and organizing their information [17].

Currently, most healthcare information systems have supported minimal security features; the problem which remains is how to enable selective information sharing without the risk of exposing additional information that needs to be protected [13]. Although many delegation models are proposed in recent years, and some of them are well applied in certain area, the applications in medical domain are limited. Let alone a comprehensive delegation architecture which could be applied in any environment.
1.3 Objectives of the Project

Based on the issues listed above, a flexible and secure delegation model is proposed. The model is focused on the delegation issues surrounding user-to-user delegation. And the objective of our model is to fulfill most delegation tasks in a flexible and cost-effective manner.

1) A delegation level decision making function is introduced so that delegations could be carried out at both role and permission levels. In other words, delegation decision would be made dynamically according to the context constraints. Either an entire role or a set of permissions would be granted to the qualified users.

2) In addition, a dynamic delegation role structure is integrated into the model to meet the ever changing delegation requirements. By doing so, the delegation model would have the learning ability to adapt itself in different delegation scenarios.

Then the incorporated medical environment would be implemented to test the feasibility of the model, different experiments would be carried out to see whether the model is valuable and could be further popularized in the future.
1.4 Synopsis of the Thesis

This thesis describes and discusses a secure and comprehensive delegation platform for web-based access control applications. It begins by reviewing the basic knowledge of RBAC concepts and delegation notions.

In the next step, the architecture of the delegation environment is designed, which includes a detailed description of delegation model. This is done by first introducing a delegation level decision making function which can make delegation decisions according to different delegation requests. Delegation function will return delegation values so that the delegation role could be easily chosen before delegation rights are granted. Then, a dynamic delegation role structure is designed to support the delegation function. By doing this, delegation roles are not constant; they could be created, deleted and marked during the delegation procedures.

After the design of the model, it is important to implement a delegation environment and apply the model to test its functionality. So an incorporated delegation environment based on an open source software is implemented. And it is necessary to consider some real world contexts and constraints that could happen to a medical environment. Next, different delegation experiments are simulated; various delegation requests will be tested.
After that, results will be collected and analyzed to compares delegation in performances in different situations.

1.5 Outline of the Thesis

The rest of the thesis is organized as follows. Related work is presented in Chapter 2. First, there is an introduction of some related delegation models which are helpful to the proposed model. Second, a description of relevent delegation applications used in medical domain which could be served as comparisons to the delegation model proposed in the thesis will be given. Chapter 3 defines and explains the model in detail from a general prospective and illustrates how the delegation model is applied in the medical domain. Chapter 4 describes the implementation and evaluation of the model in an open source platform. Conclusions and future work are presented in Chapter 5.
Chapter 2

Literature Review

2.1 Outline

This chapter consists of three parts. The first part describes the basic concepts and notions in RBAC and delegation, which is important to help understand this study. The second and the third parts present the related work. Important delegation models and delegation applications in healthcare domain are described. Also, a brief introduction and comparison of some famous open source access control platforms for healthcare information system is presented.

2.2 The NIST RBAC Model and Delegation

A study by the National Institute of Standards and Technology (NIST) demonstrates that RBAC addresses many different needs in commercial and government sectors [18]. Access control requirements were found to be determined by the need for the customers,
stakeholders, and insurer confidence; personal information privacy; the prevention of unauthorized financial asset distribution and unauthorized long-distance telephone calls; and an adherence to professional standards. Also, role-permission relationships can be predefined, which makes it simple to assign the users to the predefined roles [19] [20].

The NIST model uses a limited set of concepts to define an RBAC system. The system has users and they are assigned to roles. Each role consists of permissions and the permissions are based on the objects and operations. The following is a list of the original RBAC96 components [20] [21].

- **U** and **R** and **P** are there different sets, which respectively the signify users, roles, and permissions respectively.

- **UA ⊆ U × R**: User assignment, which is a many-to-many user to role assignment relation.

- **PA ⊆ P × R**: Permission assignment, which is also a many-to-many relation.

- **Users: R → 2^U**: A function derived from UA mapping each role r to a set of users where Users(r) = \{U | (U, r) ∈ UA\}.

- **Permissions: R → 2^P**: A function derived from PA mapping each role to a set of permissions where Permissions(p) = \{P | (P, r) ∈ PA\}.
This is the simplest RBAC form, which is shown in Figure 2-1 [6].

![Diagram of RBAC](image)

Figure 2-1: The simplest version of RBAC [6]

However, in the real world situation, roles are organized in a hierarchical structure according to their permissions within the organizations [22]. And a role is subdivided into senior role and junior role within the hierarchical structure of role groups. There exists an inheritance relationship where a senior role may inherit the permissions of a junior role. It is also possible to have multiple inheritances. Figure 2-2 shows the possible inheritance relationship inside a hospital.

In Figure 2-2, it is clear that senior roles are on the top, although all of them inherit permissions of the junior role, they have their own exclusive permissions prospectively. For instance, although both NEURO (neurologist) and CARDIO (cardiologist) are seniors of DOC (doctor) and inherit all the permissions of DOC, they own different specialities and have distinct permissions related to their specialites. Moreover, on the rightmost side, there is a multiple inheritances relationship. The role PCP (Primary Care Physician) inherits permissions from both a consultant and a DOC.
In this case, there must be more components added to the original RBAC model.

- **RH \subseteq R\times R**: A partial order on R called the role hierarchy or role dominance relation [1].
  - **U-S: user_session (u: USERS) \rightarrow SESSIONS**
  - **S-R: session_roles (s: SESSIONS) \rightarrow ROLES**

When a user logs in, all the assigned roles are associated with the session. A session allows for temporary changes to the role structure, i.e., take away a role for the duration of the session or add a role for the duration of the session [19].
Figure 2-3: Classical RBAC96 model [1]

Figure 2-3 shows the classical RBAC model, this figure illustrates all the important components in RBAC, and all of delegation model based on RBAC should follow the basic RBAC model.

The basic delegation concept is illustrated in Figure 2-4.

Figure 2-4: Basic delegation model [23]

When X invokes an operation op1 on a target Y, this operation triggers the invocation of op2 on Z. If X has the right to invoke op2 on Z but Y has no right to invoke op2 on Z, then X temporally delegates the necessary access rights to Y in order to invoke op2 on Z. In this case, X acts as the grantor, Y acts as the grantee, and Z acts as the
In the following, some well-known delegation models which are related to the current work are described.

### 2.3 Delegation Models in RBAC

There has been much research work to address the delegation issues. The most famous one is RBDM0 (Role-Based Delegation Model 0), which is based on NIST’s RBAC model. RBDM0 is the first attempt to model delegation of authorities which realizes a simple user-to-user delegation of roles. In particular, it formalizes the delegation model with total delegation, which means that each user in a delegation role that delegates the total package of permissions embodied in that role. Meanwhile, it defines a can-delegate relationship to control the user-to-user delegation. It also deals with other delegation issues including revocation and multi-step delegation [6] [7].

RBDM1, the successor of RBDM0, adopts the formalization in RBDM0 and extends it to support hierarchical roles. Since the new model was introduced to support hierarchical roles, it also defines different semantics that impact the can-delegate relation [25].

RDM2000 (Role-Based Delegation Model 2000), an extension of RBDM0, is proposed to support delegation in the role hierarchy and multi-step delegation. It
develops a rule-based declarative language to specify delegation policies and takes a
different approach from RBDM0 to solve the delegation issues [5] [26]. But when a
delegator wants to delegate a piece of role, none of the models above can provide a
satisfactory solution since the unit of delegation in them is “role”, and in many cases, it is
necessary and useful to delegate only a subset of the permissions from a role.

While all the models mentioned above are focused only on delegation of roles,
PBDM (Permission-Based Delegation Model) is developed to realize delegation based on
permissions. It supports partial delegation by separating role sets, and a subset of
permissions from a regular role is allowed to be delegated and a new delegation role is
created with the set of permissions. PBDM is actually a family of models which extends
RDM2000 to incorporate more features. It provides great flexibility in authority
management [12] [26]. Although PBDM is a complete model, the controlled propagation
on resources is not supported.

The RBDM model which uses sub-role hierarchies supports a variety of delegations.
For example, administrators can easily control the permission inheritance behavior by
using the restricted inheritance functionality. Roles are divided into a number of sub-roles
based on the characteristics of job functions and the degree of inheritance [27]. However,
like any other delegation model which is based on the role level, this model does not support permission level delegation.

Another extended RBDM model uses the characteristics of PBDM and the concept of sub-role hierarchies [28]. The advantages of both RBDM and PBDM models are thereby also available in this model. However, the role set in the model is divided into seven layers which add complexity to the realization.

With the development of information technology, traditional delegation models could not keep up with the needs of the ever-evolving information systems. It is a complex task to manage delegation and describe all the delegation requirements in a comprehensive model. Thus, delegation models themselves are extended to support new delegation characteristics [14]. And more and more delegation models are proposed by also taking account of specific delegation needs or organizational structures.

Based on the organizational hierarchies, the organizational supervised delegation model (OSDM) is proposed to identify users who must approve the delegation. The model targets at solving the problem in managing the complexity of the huge number of relations in traditional delegation models [29]. Event-Based Task Delegation aims to reason about the delegation events to specify the delegation policies dynamically to control and secure the delegation process. The model identifies two important issues for
delegation, i.e., allowing delegation tasks to carry out, and having a secure delegation within a workflow [30]. RBAC with delegation in a workflow context (DW-RBAC) is proposed to address delegation and revocation in the workflow based systems because these systems have been criticized as being inflexible for the lack of support for delegation [31].

Administration cost is also an important factor in the delegation process; however, many delegation models are cumbersome and hard to apply in real world scenarios even though incorporating various features.

In order to provide flexibility and reduce administration costs, the capability-based delegation model is proposed to achieve a higher level of collaboration in large-scale information systems. The approach to model delegation is to integrate a capability-based access control mechanism and map it to permissions as well as roles in each domain, and by means of the assignment of roles to capabilities, suitable permissions are automatically assigned to users [32].

2.4 Healthcare Applications

In emergency situations, certain subjects sometimes have to perform important tasks although they are usually not authorized to perform these tasks [33]. Delegation is a
viable mechanism to realize that. However, delegation is not the only way to achieve that. Recently, researchers have proposed different ways to solve the problem of lack of authorities in emergence, and most of them could be classified in two categories. The first one is delegation. Anther useful mechanism is the Break the Glass (BTG).

In BTG, usually the assumption that access permissions are known in advance, and that the rules have been set up correctly, but in real settings, errors are made and unanticipated or emergency situations may occur. This mandates that a more flexible and adaptable approach be adopted [34]. In such cases, a BTG policy can be used in order to break or override the access controls in a controlled manner [35]. This mechanism is very useful for intern doctors because instead of being denied by the policy, it grants the authorized interns to break the glass. Usually, break-glass solutions are implemented by issuing temporary accounts that comprise more powerful access rights (e. g., “root” accounts) on one hand and a more detailed logging on the other hand [36].

The federal government has issued multiple statements around security and privacy in relation to access control. One relevant document to health information is the white paper issued by the Office of the National Coordinator for Health Information Technology (ONC) titled Consumer Consent Options for Electronic Health Information Exchange: Policy Considerations and Analysis [37]. However, this white paper describes
the “break the glass” policies is not a best practice for U.S. national adoption [38] [39]. So it is not recommended in certain situations.

According to that, it is a natural way to choose delegation to solve the problem. In addition, delegation is a cost-effective way and saves a lot of efforts in administration. For example, such a feature is valuable for intern doctors because the administrator does not need to set up a bunch of intern doctor roles and handle many external accounts. Therefore, delegation has been applied to some healthcare information systems.

Based on that, the Role-based delegation framework integrates the RDM2000 delegation model and a rule-based language to specify and enforce delegation policy [13]. Such a framework addresses the issue of how to support selective information sharing in role-based systems while minimizing the risks of unauthorized access [13]. It is a well-applied healthcare information system framework in the former years.

Recently, the xDAuth model is proposed to establish a sophisticated framework for access control and delegation on an open source medical systems [40]. xDAuth leverages a trusted delegation service to serve as a decision making point for access requests. And instead of authenticating the user itself, the delegation service further redirects the user to an authentication service [40]. Through this way, the framework provides both
functionality and security for healthcare users. However, xDAuth model is not based on RBAC, so it is hard to integrate the useful features in RBAC context.

2.5 Open Source Electronic Health Record Systems

Since electronic health record systems are being designed to fit within both small and large organizations, providing a scalable access control that meets lawful and regulatory compliance is a key component [41]. Recently years, the open source electronic health record (EHR) system has been developed rapidly. Open source softwares have several advantages over closed proprietary systems. Firstly, the system is more 'future-proof', being able to withstand the changes in libraries, operating systems and hardwares. Secondly, open-source software is license free and allows everyone to benefit from any developments made by others, minimizing the costs to everyone involved [42]. We will give a brief introduction of some popular open source EHRs.

OpenEMR (Electrical Medical Record) is a web-based application that provides a set of pre-defined roles and permissions upon installation that an administrator can assign users to [41]. The present system has a common set of healthcare provider roles: Accounting, Clinicians, Physicians, Front Office and Administrators [32]. OpenEMR use PhpGACL (Generic Access Control List) module to control access privileges and allows
administrator to make modification of associated privileges if necessary [43]. However, OpenEMR does not have delegation module and it lacks of any emergency access procedures. In this case, it is not flexible enough to support intern doctors.

OpenERP (Enterprise Resource Planning) is a web application that handles the business logic and communicates with the database application [41]. It has a medical module that provides a complete electronic medical record with patient information and other medical information. In the current medical module, each user must have a local account in order to access medical object. This feature eases the security management of user operations; however, it clearly brings a lot of difficulties in providing the access to the interns and medical students [44].

OpenMRS (Medical Record System) is a collaborative open source project to develop software to support the delivery of health care in developing countries. The RBAC implementation in OpenMRS is entirely dynamic. Roles can be easily created and modified. OpenMRS grows rapidly in developing countries and currently it has gained its worldwide acceptance in over 23 countries. However, it has the same problems as OpenEMR. The lack of any emergency access procedures causes OpenEMR’s failure to meet the HIPAA (The Health Insurance Portability and Accountability Act) standards [45].
In order to implement the model, we choose OpenEMR as the platform because it is the most suitable one for our research.
Chapter 3

Delegation Architecture

3.1 Outline

At first, an introduction about a flexible role based delegation model is given in section 2. Then in the third section, details of delegation decision function is presented to show the how the model realize delegations at both role and permissions levels. The last two sections discuss other important delegation features like multi-step delegation and revocation and how those features integrated in the model.

3.2 Dynamic Delegation Role Structure

A role in RBAC is the aggregate of responsibility and authority, to which the access to the object is permitted [46]. It is an intuitive way to realize delegation based on the existing role structure. A flexible delegation scheme is desired in many cases as mentioned above. Figure 3-1 shows the proposed model with the structured role sets. In
the model, there are four layers of roles: normal roles (NR), predefined delegation roles (PDR), retained delegation roles (RDR) and temporary delegation roles (TDR). A role can be placed in one of the four layers by considering the different delegation scenarios. This leads to a partition of permission-role assignment (PA): permission-normal role assignment (PAN), permission-predefined delegation role assignment (PAPD), permission-retained delegation role assignment (PARD) and permission-temporary delegation role assignment (PATD).

Figure 3-1: A flexible delegation model with role structure
NR is the set of normal roles which can be used in both regular access role assignment and delegation cases. PDR, RDR and TDR are the set of roles which can only be used in delegation.

PDR is defined in advance in order to fulfill the common delegation requests in the system. In other words, PDR contributes to the most common delegation scenarios. For instance, in distributed-computing environments, applications or users have to share resources and communicate with each other in order to get their jobs done. And it would be useful to support the delegation scenarios in which a user assigns a subset of the available access rights to another user. However, it is time-consuming to set up the temporary delegation roles for each such request. Therefore, it is more efficient to establish a set of predefined delegation roles for the system. Furthermore, in traditional information system, e.g., healthcare information system, inexperienced personnel such as intern doctors would be delegated typical tasks to be trained to be qualified for the job. A set of PDR would come in handy to fulfill the requirement. The following is a set of formal definitions added in the RBAC model.

- **Permissions_ PDR (r):** $\text{PDR} \rightarrow 2^\text{P}$ is a function mapping a PDR to a set of permissions.
- **Permissions** _RDR (r)_: \( \text{RDR} \rightarrow 2^P \) is a function mapping a RDR to a set of permissions.

- **Permissions** _TDR (r)_: \( \text{TDR} \rightarrow 2^P \) is a function mapping a TDR to a set of permissions.

- \( \text{PAPD} \subseteq P \times \text{PDR} \)

- \( \text{PARD} \subseteq P \times \text{RDR} \)

- \( \text{PATD} \subseteq P \times \text{TDR} \)

- \( \text{PAD} = \text{PAPD} \cup \text{PARD} \cup \text{PATD} \)

In computer security domain, information is typically classified according to their security characteristics. Likewise, as shown in the following Table 3.1, we categorize permissions into three categories based on the information they have access to.

**Table 3.1: Categorization of permissions and characteristics**

<table>
<thead>
<tr>
<th>Degree</th>
<th>Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconditional</td>
<td>Permissions which have access to confidential data</td>
</tr>
<tr>
<td>Conditional</td>
<td>Permissions which have access to secret data</td>
</tr>
<tr>
<td>Restricted</td>
<td>Permissions which have access to top secret data</td>
</tr>
</tbody>
</table>
Normally, a PDR is the sub-role of an NR based on the job functions. A PDR contains conditional permissions. With that, a PDR can be readily assigned to any member in a certain department.

However, the number of PDR should be limited. First of all, the number of conditional permissions of a certain role is limited. Second, in RBAC, a senior role inherits all the permissions from all its junior roles. If a PDR is derived from a senior role, chances are that the PDR is identical with one of its junior roles. In case of that, there is no need to set up a PDR role. Such situation is frequent to see in large organizations and corporations, especially those with complicated inheritance relationships. Table 3.2 shows a possible scenario where a PDR and a junior role Assistant Doctor own the same permissions if the PDR is set up improperly in a healthcare information system. In particular, the permissions include create/view electronic patient records (EPR) and view prescription files (PR).

Under this circumstance, Physician inheritances all the permissions of Assistant Doctor, PDR might be identical to role Assistant Doctor if the PDR is set up inappropriately.
Table 3.2: Permissions of physician, assistant doctor and PDR

<table>
<thead>
<tr>
<th>Roles</th>
<th>Physician</th>
<th>Assistant Doctor</th>
<th>PDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create EPR</td>
<td>°</td>
<td>°</td>
<td>°</td>
</tr>
<tr>
<td>View EPR</td>
<td>°</td>
<td>°</td>
<td>°</td>
</tr>
<tr>
<td>Edit EPR</td>
<td>°</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Delete EPR</td>
<td>°</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>View PF</td>
<td>°</td>
<td>°</td>
<td>°</td>
</tr>
</tbody>
</table>

Figure 3-2 illustrates a possible simple role hierarchy and their relationships. Vertically, the senior role holds all the permissions of the junior role, while horizontally, PDR 1 is the sub-role of the Senior role and PDR 2 is the sub-role of Junior role. It is possible that PDR 1 and the Junior role have the same permissions.

Figure 3-2: Example of PDR roles in the role hierarchy
For infrequent delegation requests, the TDR will be established. In our model, all the TDR roles will be stored in a dynamic buffer. It is used to deal with short-term delegation requests by realizing permission level delegations. When there is no suitable role to fulfill the delegation request, a TDR will be created using the PATD function. A temporary delegation role would be transformed into a retained delegation role (RDR) automatically once it has met certain usage rate and been proved secure under the surveillance of the delegation monitor. Otherwise it will be deleted after certain time period due to the lack of usage. RDR roles will be readily used to serve repeated delegation requests and therefore can reduce runtime overhead to create new TDR roles.

### 3.3 Delegation Decision Function

As discussed in Section 3.2, the proposed model provides four different delegation layers to cope with different delegation scenarios. However, there is a need to provide a systematic method to identify and select an appropriate layer from the role structure to fulfill the delegation requests. Therefore, Role_of_Delegation function is introduced, which is in charge of selecting the appropriate delegation levels. It describes how the decision for the delegation level is made according to different delegation requests by
taking users, operations and objects as input, and querying the current delegation role
structure.

- **Role_of_Delegation**: SESSIONS × OPS × PERS × OBS → ROLE or NULL

- **ROLE**: There is a suitable delegation role in the role structure for the
delegation, and then the delegation will be executed at the role level.

- **NULL**: There are no suitable roles in the role structure, and in this case, the
delegation has to be performed at the permission level, which means a partial
delegation from a single or multiple roles is needed.

Figure 3-3 shows the logical flow-chart for the model. The Role_of_Delegation
function analyzes the delegation requests and interacts with the role structure to checks if
there is a suitable role for the request. If such a role exits, the function will simply return
it. Then the role could be utilized in the following delegation procedure. Otherwise, the
function returns NULL, in this case, a new TDR will be created and assigned with the
permissions specified in the request. By this way, both role level and permission level
delegations can be realized. However, permissions level delegation is more time
consuming, after a new TDR is created, it would be saved in the TDR database and
marked with usage frequency, so that it could be used in the role level delegation for a
couple of times.
3.4 Multi-step Delegation

Multi-step delegation is also an important delegation feature as it allows the delegated role memberships to be further delegated to other roles. Delegation depth denotes how many times the task has been delegated in a delegation chain. In multi-step delegation, the depth of the delegation chain is greater than one, and delegated permissions can be further delegated multiple times. Multi-step delegation is useful in a
variety of organizations. For example, in healthcare information systems, unskilled delegates like intern doctors or assistant doctors might not be able to handle the task independently, so the delegation has to be further delegated.

In traditional delegation models, permissions assigned to the delegation role are constant in the delegation chain. For example, Alice assigns a delegation role which could be her role or sub-role to Bob, and Bob further delegates the same role to Cathy. Although this arrangement simplifies the management, it is not flexible to handle certain delegation requests. For instance, a physician pays a visit to another hospital and delegates his/her role to the assistant. Later on, the assistant finds himself overloaded with other work and cannot fulfill the all the tasks alone, so he further delegates the role to an intern doctor who has time and skills to complete the task. However, due to the security policies, an intern doctor could not possess some of permissions. Usually, such a delegation request will be rejected and the assistant has to find another user to further delegate the entire privileges. However, it is possible that the delegation request be rejected again. In emergency situations, it will bring obstacles in carrying out the delegation in a timely manner.

In this model, the delegation chain is more flexible. The delegator may modify the delegation role to further assign it to the next delegatee. However, the subsequent
delegator can only remove some of the original delegation privileges and further assign it to other users. For instance, Alice delegates her role to Bob; therefore a delegation chain is created. Later on, Bob further delegates the role to Eric; he can only re-delegate the role or a part of role. Such a scheme would be able to address the problem raised in the previous example, and in that case, the assistant is able to only delegate some of the physician’s permissions to the intern and keep the rest; and the whole operation can be performed in a timely manner. Moreover, such a scheme is easy to implement in our model. Once the original delegation role is modified in the delegation chain, the proposed role structure and decision function would collaborate to make the appropriate delegation choice.

### 3.5 Delegation Revocation

Since the proposed model supports delegation at both role level and permission level, multiple revocation schemes are supported to make the model robust and flexible, including timeouts and requiring the original member of the delegating role to revoke the delegation. Meanwhile, it is also possible to execute the revocation by revoking a user from the delegation role or removing a subset of permissions from that delegation role.
Furthermore, due to the structural delegation role sets, revocation schemes are also layered. Simple revocation schemes lay at the bottom level, while fine-grained control revocation schemes sit at the top. For instance, when a delegation role PDR is performing, it corresponds to a relaxed revocation scheme because it is considered as relatively secure. In this case, multiple revocation schemes are supported, such as grant-dependent revocation in which only the delegator is allowed to revoke the delegated privileges from the delegatees. In the meantime, noncascading revocation is supported which corresponds to multi-step delegation, in which the delegation between the second and the third user will not be revoked even when the first user revokes the delegation. For instance, in the multi-step delegation scenario, a PDR is delegated from Alice to Bob, and eventually transferred to Eric, the noncascading revocation is allowed in such a case. The revocation steps would be as follows.

1) Select the PDR which is delegated by Alice.

2) Remove the PDR in the first step of the delegation, in order to revoke the roles assigned to Bob.

3) The delegation chain from Alice to Bob is revoked.

4) Select the PDR which is delegated by Bob.
5) Remove PDR in the second step of the delegation, in order to revoke the roles assigned to Eric.

6) The delegation chain from Bob to Eric is revoked.

7) The entire delegation chain is revoked.

Even though Alice and Bob use the same PDR, the delegation chain has to be revoked step by step.

Noncascading and cascading revocation are both allowed if a PDR is used in the delegation. However, when the delegation request is from outside of an organization and a TDR is used, it would be considered less secure and the delegation should be under higher security supervision and demand a stricter revocation. Normally, once there are some abnormalities or when violations are detected, any user who has the privileges can revoke the delegation and the entire delegation role would be revoked. In this situation, if the delegation between the first user and the second user is revoked, the entire delegation chain will be revoked. Noncascading revocation is not allowed in delegations with a TDR. The revocation steps would be as follows.

1) Select the new created TDR which is delegated by the original delegator.

2) Remove the permissions associated with the TDR, which means to revoke the PATD function.
3) Remove TDR so as to revoke the roles assigned.

4) The entire delegation chain is revoked.
Chapter 4

Implementation and Evaluation

4.1 Outline

In this chapter, the implementation of the flexible delegation model based on a healthcare information system is presented. Details of how to apply the dynamic delegation role structure in healthcare information system is given. Then, based on that, different tests are carried out to verify the feasibility and flexibilities of the delegation model. Moreover, evaluations are presented to discuss whether such a model could be applied in the healthcare domain.

4.2 Implementation

SanyYeob Na has proposed the guideline of how to realize delegation in information systems [23]. The first step is to build a delegation server. Delegation server makes a decision about whether the delegation process is permitted or not and maintains some of
delegation information for making a decision [23]. We followed the guideline in designing our delegation system and its architecture is shown in Figure 4-1.

Figure 4-1: The delegation system
Delegation server is the core component in the implementation. On the one hand, it accepts delegation requests, conducts checks and controls the execution of the delegation procedure based on the result of Role_of_Delegation function; on the other hand, it receives the feedbacks from Delegation monitor at the end of the delegation session, which includes useful information like whether the delegation is successful or not, what kinds of errors in case of failure, and so on. The feedbacks will be used to optimize the selection of delegation levels, delegation roles and delegatees. Each time the delegation task is fulfilled, the information about the delegation roles used in the process would be automatically stored or updated in the role set database.

The delegation components are based on OpenEMR, which provides a set of pre-defined roles and permissions upon installation. OpenEMR is built in PHP languages and the source code is available in its website [43]. However, OpenEMR does not have a delegation mechanism, so design a delegation server which controls the whole delegation process in the system is the main object.

As shown in Figure 4-1, delegation starts from user delegation request. There are multiple delegation requests been applied in different systems, and delegation request form is the most frequent one.
Figure 4-2: Delegation request form

Figure 4-2 shows the implementation of delegation request form in OpenEMR. In the delegation request form, the first four fields are used for authentication. “Role Info” is the role that the delegator currently holds. The delegator needs to specify whether he/she wants to delegate the entire role or just some of the permissions in the delegation type option. If the role delegation type is chosen, then the entire role will be delegated. If permission delegation type is selected, the delegator needs to specify the permissions and objects he/she wants to delegate. “Addonly” and “Write” are the two fields which store detailed operations associated with the objects. Upon the completion of the form, the
delegation request will be sent to the delegation server. Then the delegation server will search the current delegation role structure for suitable existing roles for delegation. A TDR will be automatically created if no such role is identified.

The delegator can customize the permissions by manipulating the “Permissions” and “Objects” fields in the request form. In addition, the delegatee can be selected. However, the default delegate selection is null, because in the advanced setting, administrator can filter the delegatees according to their own criteria. Once the delegator submits the form, the delegation request will be sent to the delegation server for further processing.

![Delegatee Selection](image)

**Figure 4-3: Delegatee selection**

In order to facilitate the delegation administration, a delegation selection page as
shown in Figure 4-3 was implemented. This figure shows all the employees who have been chosen as delegatees and fulfilled the task at least once. The administrator could make a choice of the delegatee according to the delegator’s needs.

As discussed, the structured delegation role layers are helpful when dealing with the ever-changing delegation environment. In healthcare domain, if the delegation request comes from inside a hospital, PDR and RDR would come into play in most of the cases, because PDR and RDR are considered the more frequently used in the delegation scenarios. When there is a one-time delegation request from outside the hospital, for example, a clinic demands to have certain permissions for a patient’s record, TDR is more appropriate than PDR and RDR. This delegation leads to uncertainty and has the involvement of a security administrator. In this scenario, a temporary delegation role will be created in the delegation role set, which is anonymous and could be abandoned once the delegation is fulfilled.

TDR is also very useful for intern doctors. During the internship, different intern doctors could be assigned with different tasks. Especially in the intermediate stage of internship, Intern doctors would be allocated to different departments and physicians. Therefore, their individual duties and responsibilities differ from each other. The
attending physicians would assign specific tasks for each intern doctor and control their authorities. In this scenario, the physician can define their own TDRs which incorporate part or entire permissions of the physician. E.g. if two interns are allocated into a PCP and a Specialist Physician’s office respectively to carry out different tasks, the PCP and Specialist Physician could assign their own specific permissions to their interns.

To apply the dynamic delegation role structure, first of all, PDR set has to be set up. Some basic permissions are summarized to define a few PDRs for every hierarchy in the OpenEMR; meanwhile, a PDR intern doctor is also defined which contains the least privileges in the hierarchy. The reason for doing so is because OpenEMR does not provide any role or permission for this special group. Figure 4-4 lists some of the permissions provided by OpenEMR.

<table>
<thead>
<tr>
<th>Acl_id</th>
<th>Section_value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Acct</td>
<td>bill</td>
</tr>
<tr>
<td>10</td>
<td>Admin</td>
<td>acl</td>
</tr>
<tr>
<td>12</td>
<td>Patients</td>
<td>docs</td>
</tr>
<tr>
<td>15</td>
<td>Placeholder</td>
<td>filler</td>
</tr>
</tbody>
</table>

Figure 4-4: Selected permissions from OpenEMR system
4.3 Programming and Database

OpenEMR uses phpGACL module to control the authorities of different roles, phpGACL is a set of functions that apply access control to arbitrary objects (web pages, databases, etc.) by other arbitrary objects (users, remote hosts, etc.) [47].

Figure 4- 5: PhpGACL component to control access rights

PhpGACL offers fine-grained access control with simple management, and is very fast. In addition, it is written in PHP. Figure 4-5 is the phpGACL component imbedded in
OpenEMR and it shows how to control access rights in phpGACL. There are some definitions here.

- **ACO**: Access control object, which is the subject we want to control the access to (e.g., the patient’s medical record).

- **ARO**: Access require object, which is actually the user who requires an access.

PhpGACL provides many APIs to control the access to certain object, so that it is natural to use those APIs to authenticate user identity and authority permissions through delegation.

Database is also indispensable in the realization of the model. OpenEMR uses MySQL as the backend database because the combination of php and MySQL enables programmers to create applications that will run on just about any platform.

Once we create a new role, the role is stored in table “users” in OpenEMR. Table 4.1 is part of the “users” table:

<table>
<thead>
<tr>
<th>ID</th>
<th>Username</th>
<th>Password</th>
<th>Specialty</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Zliu7</td>
<td></td>
<td>admin</td>
</tr>
</tbody>
</table>
Meanwhile, as discussed, openEMR uses PhpGACL to realize access control. Some specific user information would be stored in PHPgacl module. In table “gacl_aro”, user information is stored. Table 4.2 is part of the “gacl_aro” table:

<table>
<thead>
<tr>
<th>ID</th>
<th>Section_value</th>
<th>Value</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Users</td>
<td>Admin</td>
<td>Administrator</td>
</tr>
<tr>
<td>12</td>
<td>Users</td>
<td>Zliu7</td>
<td>Andy Liu</td>
</tr>
</tbody>
</table>

Correspondingly, the role information was stored in table “gacl_aro_groups”, which specified all the roles used in the system. To map the user and role relationship, the table “gacl_groups_aro_map” realizes that using the above two tables.

Each role has certain permissions, but in PHPgacl, permissions are categories in access control list and each category corresponds to specific permissions. Table “gacl_acl” stores all the categories. Table 4.3 shows the control list for different roles. As mentioned in the delegation request form, permissions are granted through “write” and “addonly”.

Meanwhile, there should be a table “gacl_aro_groups_map” relates each role to the access control list, just like table “gacl_groups_aro_map”, it also contains two columns.
Table 4.3: Permission information controlled by phpGACL

<table>
<thead>
<tr>
<th>ID</th>
<th>Return Value</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>write</td>
<td>Administrators can do anything</td>
</tr>
<tr>
<td>11</td>
<td>addonly</td>
<td>Objects that physicians can read but not modify</td>
</tr>
<tr>
<td>12</td>
<td>write</td>
<td>Objects that physicians can read and modify</td>
</tr>
</tbody>
</table>

In summary, table “gacl_aco” stores all the specific permissions and table “gacl_aco_map” relates the access control list to each of the permissions.

In order to realize delegation service for the proposed model, first of all, a table that the role can relate permission directly would be needed. Meanwhile the table should contain a field that specifies the permissions that a role contains, as depicted in Table 4.4. When a delegation request arrives, it is easy to query the table whether there is a suitable candidate role which contains identical permissions specified in the request.

Table 4.4: Role_Permission_Map

<table>
<thead>
<tr>
<th>Group_id</th>
<th>Aco_id</th>
<th>Num_aco</th>
</tr>
</thead>
</table>

Moreover, in delegation we need a table to calculate how many times a role has been delegated successfully. We only need this for temporary delegation role. Table 4.5 calculates the successful delegation rates for TDRs.

Table 4.5: Rates of fulfilled TDRs

<table>
<thead>
<tr>
<th>Group_id_TDR</th>
<th>Sus_del_rate</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
</table>
And for the user, we need to calculate the successful delegation rate and unsuccessful rate for each user. This information is important for administrator to select the best delegatee in different situations. Table 4.6 records such information.

Table 4. 6: Information of successful and failed delegation

<table>
<thead>
<tr>
<th>Aro_id</th>
<th>Sus_del_rate</th>
<th>Unsus_del_rate</th>
<th>Name</th>
</tr>
</thead>
</table>

4.4 Evaluations

In this Section, different scenarios in user-to-user delegation are simulated based on the prototype system to evaluate the functionality and performance of the model. Initially a predefined delegation role Physician_intern is set up, which will be used when a physician delegates some permissions to an intern doctor. The Physician_intern role consists of permissions (Documents (read), Medical history (read/write)).

Suppose that Alice is a physician in the hospital. And she wants to delegate some permission to an intern doctor Bob by sending a delegation request which contains the permissions (Documents (read) and Medical history (read/write)). The authorization check will pass as there are no security violations. After that, the Role_of_Delegation function returns Physician_intern, which is exactly the desirable delegation role. Then the
delegation session starts. This delegation operation is named DO1 to compare the results in the following stages.

Likewise, the delegation role set is extended by adding two more predefined delegation roles including Surgeon_assit and Surgeon_intern.

Then another physician Bill wants to delegate a part of his permissions (Documents (read/write) and Medical history (read/write)) to the assistant Dan, he simply sends a delegation request message to the delegation server. Then the Role_of_Delegation function returns NULL, which means there is no feasible delegation role under current circumstances. In this case, the system will create a temporary delegation role in which permissions the physician referred to are incorporated to perform the delegation task. When the physician’s assistant logs in, the delegation module will notify Dan that the delegation role with the necessary permissions have been assigned in order to fulfill the delegation task. When the delegation session is expires, the delegation monitor will check if the task is fulfilled successfully. After that, the monitor will send a feedback to the delegation server and the temporary delegation role TDR1 will be saved in the temporary delegation role buffer. Likewise, this delegation operation in this stage could be called DO2.
Later, the same delegation request is issued by Alice to delegate the same set of permissions to her assistant, this time the Role_of_Delegation function returns TDR1, which indicates that the saved temporary delegation role which is stored earlier in the role structure will work appropriately. In the prototype implementation, there is a delegation frequency for each delegation role. A temporary delegation role will become a retained delegation role once the frequency reaches 10. In the evaluation, the TDR1 will be performed for sufficient times so that it turns into a retained delegation role (RDR1). In this case, Role_of_Delegation returned RDR1 which is deemed to be more trustworthy.

This time, the delegation operation is referred to as DO3. Table 4.7 shows the changes of the role structure between each of the delegation operations.

<table>
<thead>
<tr>
<th>Delegation Operations</th>
<th>Number of Roles</th>
<th>Request Fulfilling Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NR</td>
<td>PDR</td>
</tr>
<tr>
<td>DO1</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>DO2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>DO3</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4. 7: Performance evaluation
Initially, there are only 6 normal roles which are set up in advance, so the number of NR is constant. The numbers of RDR and TDR are dynamically changing due to the change of the delegation frequency. This evaluation shows the flexibility of the delegation model because it can make appropriate delegation choices according to different delegation requirements. Meanwhile, the retained delegation role set in our model is dynamic as one temporary delegation role could be transformed into a retained delegation role while another could be removed from the temporary delegation role structure. By doing so, the model is adaptive to the ongoing delegation requests.

Then it is time to evaluate the multi-step delegation. This time Bill delegates the same task to his assistant Dan. However, Dan is not able to finish the task and he further delegates the task to Bob. In both steps of the delegation, Role_of_Delegation returns RDR because there are no modifications in the delegation requests. After the user assignment indicates that the delegation will be performed at role level, the delegation session starts.

In addition, a performance comparison between our model and a simple PBDM model is conducted, the simple PBDM implementation is based on the prototype system. However, the Role_of_Delegation is disabled so that delegation could only be performed at the permission level. In the evaluation, the number of roles is gradually increased to
compare the efficiency between the two models. Simple delegation scenarios are monitored exactly as mentioned above and recorded the request fulfilling time which is the total elapsed time to fulfill the delegation request.

The results show that the request fulfilling time for PBDM is almost constant because each time a delegation contains a piece of permissions, a new temporary delegation role will be created, and the time is not affected by the number of roles in the role structure. The proposed model is slower than PBDM initially, because the number of roles is limited and most of the delegations had to be performed at the permission level, and our model had to call the Role_of_Delegation function for the delegation decision making. As the number of roles increases, the request fulfilling time of proposed model decreases because more and more delegation requests are being performed at the role level. The evaluation shows that our model is efficient and adaptive in typical delegation scenarios in a healthcare information system, and the model can help to accelerate the delegation decision making process by identifying the suitable delegation levels and roles automatically.
Chapter 5

Conclusion and Future work

5.1 Summary and Conclusions

In the field of access control, in order to simplify the administrator task and to manage collaborative work securely, it is important to have a system that allows delegation, especially with the increase of shared information and distributed systems. Thus, a comprehensive access control model must provide a flexible administration model to manage delegation and revocation [14].

In this thesis, a flexible delegation model is proposed in an effort to fulfill a variety of delegation scenarios. Meanwhile, the delegation role decision function, as the key component in the model, enables delegation requests to be fulfilled at suitable levels automatically. The model has been shown to be effective and efficient through a prototype implementation and evaluation in a healthcare information system.
5.2 Future Work

There is still much room to further improve the current work. First of all, there should be more experiments to be carried out based on the prototype platform. Other important delegation features, such as multiple delegation and grant-independent revocation should be verified to test the feasibilities of the model. Second, the performance evaluation shall be enhanced by deploying full-fledged RBDM and PBDM models or other important delegation models.

In addition, the current model is only based on user-to-user delegation. However, role-to-role delegation is also frequent in our daily lives. For the next step, the model should be extended to support the role-to-role delegation. Furthermore, evaluations based on a distributed healthcare information system shall be developed and performed to gain more insights of the proposed model.
References


[23] SangYeob Na, SuhHyun Cheon, “Role Delegation in Role-Based Access Control,”
Proceedings of the Fifth ACM Workshop on Role-Based Access Control, 39-44, July
26-28, 2000, Berlin, Germany.

Based role Framework for Access Control,” Proceedings of the First ACM Workshop
on Role-Based Access Control, November 30-December 02, 1995, Gaithersburg,
Maryland, United States.

(RBDM1),” Proc. The 20th Annual Computer Security Applications Conference,

[26] Longhua Zhang, Gail-Joon Ahn, and Bei-Tseng Chu, “A Rule-Based Framework for
Role-Based Delegation,” Proc. The 6th ACM Symposium on Access Control Models

[27] HyungHyo Lee, YoungRok Lee, and BongNam Noh, “A New Role-Based
Delegation Model Using Sub-role Hierarchies,” Proc. The 18th International Symposium
on Computer and Information Sciences (LSCIS 2003), Antalya, Turkey, Nov. 2003,
811-818.


and Technologies, June 03-05, 2009, Stresa, Italy.


  http://healthit.hhs.gov/portal/server.pt/gateway/PTARGS_0_11673_911197_0_0_18/ChoiceModelFinal032610.pdf


