Framework for Role-Based Delegation Models

by

Ezedin S. Barka
A Dissertation
Submitted to
the Graduate Faculty
of
George Mason University
in Partial Fulfillment of
the Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____ Dr. Ravi Sandhu, Dissertation Director

_____ Dr. Edgar Sibley

_____ Dr. David Rine

_____ Dr. Xiaoyang Wang

_____ Dr. Stephen G. Nash, Associate Dean
of Graduate Studies and Research

_____ Dr. Lloyd J. Griffith, Dean, School of
Information Technology and
Engineering

Date: _____ Summer Semester 2002
George Mason University
Fairfax, Virginia

FRAMEWORK FOR ROLE-BASED DELEGATION MODELS

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University.

By

Ezedin S. Barka
BS, University of Indiana, Bloomington, IN, May 1983
MS, University of Maryland, College Park, MD, December 1991

Director: Dr. Ravi S. Sandhu, Professor
Information and Software Engineering

Summer 2002
George Mason University
Fairfax, Virginia

DEDICATION

"All Praises due to Allah, the graceful, and the most sustainer"

I would like to dedicate this to my parents for bringing me to this world and for their continuous prayers and encouragement, to my wife Hadia for her patient and unlimited support, to my two sons Rabie and Rauf for bringing happiness to my life.

Without their prayers and love, this work could not have been completed.

ACKNOWLEDGEMENTS

I would like to sincerely express my gratitude and appreciation to my dissertation director, Professor Ravi Sandhu, who has been so graceful all the way and has provided valuable guidance and encouragement during my doctoral study.

Also, I would like to thank the members of my committee, Professor Edger Sibley, Professor David Rine, and Professor Xiaoyang Wang. I am thankful for their valuable comments and suggestions on my dissertation.

I am particularly thankful to my parents, to my wife Hadia, and to my two sons for their prayers and patience.

Last, but not least, I would like to thank my friends at George Mason University for their support and feedback. Special thanks to my friends Tarik Himdi and Qamar Munawar for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

FRAMEWORK FOR ROLE-BASED DELEGATION MODELS

Ezedin S. Barka, Ph.D.

George Mason University, 2002

Dissertation Director: Dr. Ravi S. Sandhu

The basic idea behind delegation is that some active entity in a system delegates authority to another active entity in order to carry out some functions on behalf of the former. Delegation can take many forms: human to human, human to machine, machine to machine, and perhaps even machine to human. In this dissertation, I focus on the human to human form of delegation. Specifically, I consider the ability of a user who is a member of a role to delegate his or her role to another user who belongs to some other role. For example, a professor in a university who is also a member in an advising committee role can delegate his/her membership in the advising committee role to another professor who belongs to another committee role. This delegation can take the form of being either permanent or temporary delegation. Moreover, the same professor can delegate only part of his/her professor role (i.e. instructor) to his/her assistant. This delegation can be only temporary.

In this dissertation, I present a comprehensive approach to role-based delegation. More specifically, I identify the characteristics related to delegation, which can be used to develop delegation models; I use a systematic approach to reduce a large number of possible cases to smaller sensible ones; and I formally define and derive some delegation models using roles based on those cases.

The thesis of this research is as follows:

It is possible, by adding a can-delegate relation to the RBAC model in conjunction with constraints, to produce a framework for role-based delegation models. The research approach used to produce a framework for role-based delegation models is an exploratory approach.

In this dissertation, the scope of my work is to address user-to-user delegation based on RBAC96. I use the RBAC96 family of models as the base for my research. I first consider temporary delegation within the framework of RBAC96-Flat-Roles (or RBAC0). Then I evolve the model to address other variations of delegation that include delegation based on role hierarchies, permanent delegation, partial delegation, delegation based on the administrator of the actual delegation, and so forth. I also address some issues that deal with revocation. In particular, I consider cascading revocation and grant-independent revocation. I chose this approach in order to work out a simple but useful

model in complete detail and then to extend this model gradually to introduce other aspects to add functionality in an incremental manner.

This dissertation shows that by adding a can-delegate relation to the RBAC model in conjunction with constraints, it is possible to produce a framework for role-based delegation models.

Chapter 1

INTRODUCTION AND PROBLEM STATEMENT

1.1. Introduction

Role-based access control (RBAC) has received considerable attention as an established alternative to traditional discretionary and mandatory access control [FCK95, SCFY96, San97]. A role is a semantic construct forming the basis for access control policy. In RBAC, permissions are associated with roles, and users are made members of appropriate roles based on their responsibilities and qualifications, thereby acquiring the permissions of these roles. In RBAC, users can be easily reassigned from one role to another, roles can be granted new permissions for new applications as systems come online, and permissions can be revoked with regard to roles as needed. This greatly simplifies security management.

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. Delegation in computers can be human-to-human, human-to-machine, machine-to-machine, and perhaps even machine-to-human. Most delegation models in the literature address human-to-machine and machine-to-machine delegation [Glad97], [ABLP96], [GM90],

[VAS91]. Models for propagation of access rights also relate to delegation indirectly (e.g. HRU, TAM, ATAM, SPM, and the Take Grant model) [HRU76], [San97], [Lamp71].

In this dissertation my focus is on human-to-human delegation. Specifically, I consider the ability of a user who belongs to a certain role to delegate a role to another user who belongs to another role. For example, a professor in a university who is also a member in an advising committee role can delegate his/her membership in the advising committee role to another professor who belongs to another committee role. This delegation can take the form of being either permanent or temporary delegation. Moreover, the same professor can delegate only part of his/her professor role (i.e. instructor) to his/her assistant. This delegation can be only temporary. The type of delegation discussed above has not received much attention in the literature so far.

This dissertation takes an exploratory approach towards analyzing the problem of delegation and producing a framework for role-based delegation models. I begin by identifying a number of characteristics related to delegation between humans, then I use these characteristics to create an exhaustive combination of possible delegation cases, and lastly, I develop a framework for building good cases that can be used for developing potential role-based delegation models. This is the first systematic attempt toward addressing the problem of delegation between humans using roles. I emphasize that the delegation itself occurs within the computer system even though it is human to human.

This work involves the investigation and formalization of role-based delegation models using nine different delegation characteristics. These nine characteristics give us a large number of possible combinations. I systematically reduce this large number of possibilities to a few practically useful cases. These cases are used for formalizing the aspects of delegation based on the Role-Based Access Control Model (RBAC) [San96].

To appreciate the motivation behind role-based delegation, consider the roles in Figure 1.1 from a hypothetical computer science department in a university. An intuitive physical scenario to illustrate delegation would be to have a professor give the key for his office to a secretary to do some filing, or allow his teaching assistant to administer an exam or grade homework. Another scenario would be to have a guest speaker from outside of the school faculty substitute for the original assigned professor. All of these activities are considered delegation simply because in each case an original member of a role is delegating his/her role membership to someone else to perform some task on his or her behalf. This can benefit the overall interests of the organization by letting the work continue even in the absence of the original member of that role. These types of activities have to be monitored and controlled in such a manner so that the resources inside the organization can stay protected. For example, in Figure 1.1 a professor could be permitted to delegate the professor role to a secretary or a teaching assistant but not to a student. Also, for example, a teaching assistant who is given the key to a professor's

office is not allowed to further give the key to someone else (this is called one step delegation).

```
┌─────────────────────────────────────────────┐
│  Secretary Chairman    Professor             │
│  ┌─────────┐ ┌─────────┐ ┌─────────┐         │
│  │ Sandy   │ │ Alice   │ │ Bob     │         │
│  │         │ │         │ │ Charlie │         │
│  └─────────┘ └─────────┘ └─────────┘         │
│  Student    Teaching Assis.  Doc. Student    │
│  ┌─────────┐ ┌─────────┐ ┌─────────┐         │
│  │ Ali     │ │ Ezedin  │ │ Gail    │         │
│  │ Bob     │ │ Qamar   │ │ June    │         │
│  └─────────┘ └─────────┘ └─────────┘         │
└─────────────────────────────────────────────┘
```
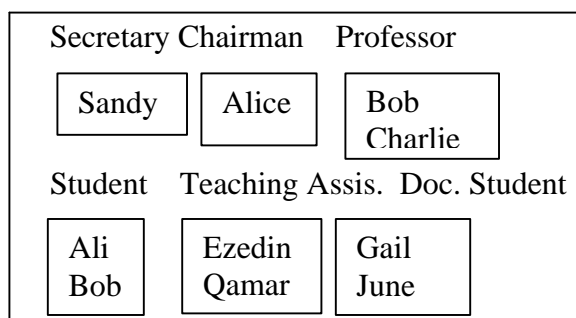
Figure 1.1. Computer Science Department Roles

1.2   Brief Overview of Role-Based Access Control Model (RBAC96)

The RBAC96 model, which was developed by Sandhu, *et al.* [San96], is based on three sets of entities called Users (U), Roles (R), and Permissions (P) (see Figure 1.2).

A user (U) is a human being or an autonomous agent. A role (R) is a job title or a job function in the organization with associated semantics concerning responsibility and authority. A permission (P) is a description of the type of authorized interactions a subject can have with one or more objects.

Access control policy is embodied in RABC components such as user-role, role-permission, and role-role relationships. These RBAC components determine whether a particular user is allowed access to a specific piece of system data. A user can be assigned many roles, and a role can be assigned to many users. The many-to-many assignment relation User -Assignment (UA) captures this property. A role can be assigned many permissions, and permission can be assigned to many roles. The many-to-many assignment relation Permission -Assignment (PA) captures this property.
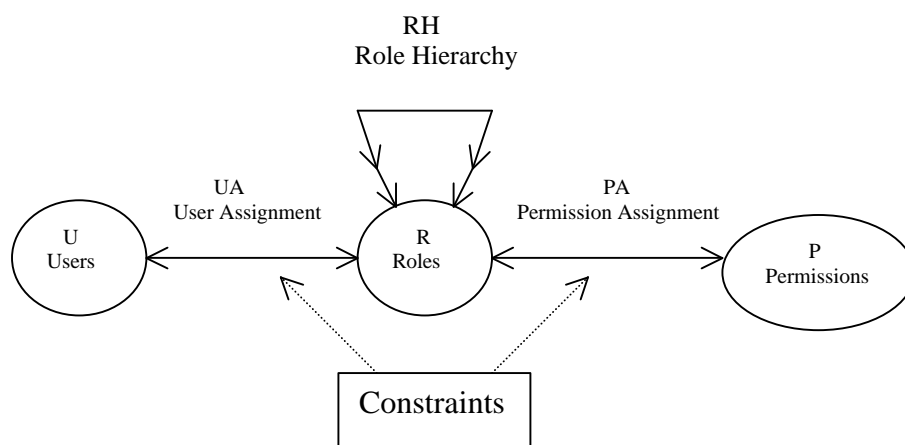
Figure 1.2. Simplified Version of RBAC96 Model

The formal definition for RBAC96 is as follows:

Definition 1.1: The RBAC96 model has the following components:

1. U, R, P, which are, respectively, the sets of users, roles, and permissions.

2. UA ⊆ U x R, which is a many-to-many User-Assignment relation assigning a user to roles.

3. PA ⊆ P x R, which is a many-to-many, Permission-Assignment relation assigning permissions to roles.

4. RH ⊆ P x R is a partial order on R called role hierarchy.

I have omitted the session concept from RBAC96 for simplicity, since it is not directly relevant to the work in this dissertation.

1.3   Problem Statement

In the information security arena, one of the most interesting and promising techniques proposed is Role-Based Access Control. In the last few years, much work has been done in the definition and implementation of RBAC. However, so far the concept of delegation in RBAC has not been studied. Delegation in computer systems can be human to human, human to machine, machine to machine, and perhaps even machine to human. These types of delegations have received some attention in the literature; however, the concept of human to human delegation has not been systematically analyzed. This thesis focuses on human to human delegation in computer systems. Specifically, I develop a series of simple but practically useful models for delegation, in which a user can use RBAC philosophy to delegate his or her role to another user who belongs to another role. This research is the first attempt to address this type of delegation.

Performing human to human delegation within the framework of RBAC will contribute to the evolution of RBAC, adding to the already positive reputation of RBAC, and will give us a simple and effective way to address the concept of delegation between humans.

The scope of my work, then, is to address user-to-user delegation based on RBAC. I will use the RBAC96 family of models as the base for my research [SCFY96]. I will first consider temporary delegation within the framework of RBAC96-Flat-Roles (or RBAC0). Then I will evolve the model to address other variations of delegation that include delegation based on role hierarchies, permanent delegation, partial delegation, delegation based on the administrator of the actual delegation, and so forth. I will also address some issues that deal with revocation. In particular, I will consider cascading revocation and grant-dependent revocation. I chose this approach in order to work out a simple but useful model in complete detail and then gradually to introduce other aspects to add functionality in an incremental manner.

1.4   Thesis

The thesis of this research is as follows:

It is possible, by adding a can-delegate relation to the RBAC in conjunction with constraints to produce a framework for role-based delegation models.

The research approach used to produce framework for role-based delegation models is an exploratory approach.

1.5   Summary of Contributions

(1)   My first contribution in this dissertation is that I have provided a framework for role-based delegation models. This was accomplished by identifying the characteristics related to delegation, using these characteristics to generate possible delegation cases, and using a systematic approach to reduce the large number of cases into a few cases, which can be used to build role-based delegation models.

(2)   My second contribution is that I derived two role-based delegation models to illustrate some practical access control policies. These policies were illustrated both in flat and hierarchical roles. I showed that using a relation, called can-delegate, a user who belongs to a certain role can delegate his/her role to another user who belongs to another role. I also showed that the delegated role can be revoked.

I have analyzed the different properties of the can-delegate relation, with each model individually as well as with the two models combined. In the case of the

flat relation between the roles, can-delegate is straightforward, involving only two

roles. However, in the hierarchical relations, can-delegate is more complicated --

it involves the two roles as well as their junior roles.

1.6   Organization of the Dissertation

Chapter 2 gives an overview of the related work. In particular, it reviews the literature

and breaks the prior work related to delegation into two types: the work directly related to

delegation and the work indirectly related to delegation.

Chapter 3 explains in detail the actual framework for role-based delegation models. It

starts by identifying the characteristics related to delegation, showing how these

characteristics are used to generate possible delegation cases, and uses a systematic

approach to reduce the large number of cases into a few useful cases that can be used to

build practical delegation models.

In the next two chapters (4 and 5) I introduce and derive role-based delegation models to

illustrate some practically useful access control policies. Chapter 4 explains the Role-

Based Delegation Model with Temporary Self-Acted Delegation (TRBDM). This is

addressed in two sections: 4.1 addresses the Role-Based Delegation Model in Flat Roles,

while section 4.2 addresses the same model in roles that have a hierarchical relationship.

Chapter 5 explains the Role-Based Delegation Model with Permanent Delegation

(PRBDM). Similar to the case of TRBDM, PRBDM also addresses the delegation in two sections, in flat roles, and in hierarchical roles respectively. Finally, Chapter 6 concludes this dissertation. It enumerates the contribution of this dissertation and discusses future directions.

Chapter 2

RELATED WORK

2.1   Work Directly Related to Delegation

Gasser and McDermott [GM90] defined user to machine delegation as "the process whereby a user in a distributed environment authorizes a system to access remote resources on his behalf." The user's authorization of this process to act on his behalf is a form of delegation of rights from the user to the process. In some cases the user may delegate the rights to one of several permissible roles or identities (e.g., by logging in using different names and/or passwords) in order to limit the actions of the process to some subset to which the user is authorized. Limited delegation also occurs routinely in multi-level secure systems where the user selects a single classification of his process that is a subset of the access class for which the user is authorized [GM90].

In their paper, Gasser and McDermott described a technique for delegation that provides both cryptographic assurances that a delegation was authorized and authentication of the delegated system, thereby allowing reliable access control as well as precise auditing of the system involved in every access. They claim that their technique goes further than other approaches for delegation in that it also provides termination of a delegation on

demand (as when the user logs out) with the assurance that the delegated systems, if subsequently compromised, cannot continue to act on the user's behalf. Delegation and revocation are provided by a simple mechanism that does not rely on online trusted servers.

Gladny [Glad97] considered the security requirements for a digital library that emulates massive collections of paper and other physical media for clerical, engineering, and cultural applications. He described an access control method that mimics organizational practice by combining a subject tree with ad hoc role granting, that controls privileges for many operations independently, that treats privileged roles such as auditor and security officer like every other individual authorization and that makes access control information part of ordinary objects. This access control method scales efficiently from a very small number to a large number of users. The method is accomplished by emulating vertical delegation in organizational hierarchies, extended to permit privilege delegation from any one node to any other node, up, down, or across the organization tree. This provides a way to represent special administrative roles like security officers. A driving objective is that every privilege should be traceable as a sequence from a custodial user to its holder.

Varadharajan, et. al. [VAS91] considered the problem of delegation of right, or proxy, in distributed systems, whereby some objects may be authorized to act on behalf of other objects. They considered the essence of the delegation problem to be the verification that

an object that claims to be acting on another's behalf is indeed authorized to act on its behalf (object to object delegation). In practice this means that we need to ensure that the information is securely transferred between the objects.

In his paper, Varadharajan presented a signature-based scheme to achieve delegation that requires different inter-object trust assumptions. These schemes can be instantiated using public key and secret key based cryptographic techniques. In both public key and secret key based approaches, protocols for delegation were proposed, along with possible alternatives to those protocols.

In his paper, Varadharajan also discussed revocation, the facility whereby a delegator can withdraw individual delegations. He claims that such a facility is important for limiting damage, which may occur as a result of delegation. He considered several mechanisms for implementing revocation. These considered mechanisms were compared to the mechanism proposed in the Distributed System Security Architecture described by Gasser and McDermott [GM90].

Finally, Varadharajan looked at the Kerberos authentication for client/server distributed systems and determined that the mechanisms in Kerbaros are not entirely suited to delegation. He thus proposed extensions to the Kerberos mechanism to implement delegation.

Hagstrom, Jajodia, Presicce, and Wijesekera [HJP01] provided a comprehensive study of the problem of revoking a chain of granted accesses that are formed as a result of granting access and administrative rights in an ownership-based framework for access control and the impact different revocation schemes may have on the chains. They have argued that several different semantics are possible for the revoke operation, and that the appropriate semantics depend on the particular situation. They have identified three main revocation characteristics: the extent of the revocation to other grantees (propagation), the effect on other grants to the same grantee (dominance), and the permanence of the negation of rights (resilience). They devised a classification using these three dimensions. They used these dimensions to define eight different revocation schemes. The framework that they used to describe the semantics was general enough to let the revocation schemes be incorporated in other systems. Also, they have shown that the schemes proposed in the literature can be classified within their framework, and they claimed that -- depending on the context -- designers of access control systems can use one or several of their schemes to implement revocation functionality.

Sandhu, Bhamidipati, and Munawer [San, Bha, and Mun 97] described the motivation and intuition and outlined a new model for RBAC administration called ARBAC97 (administrative RBAC '97). ARBAC97 has three components: URA97 (user-role assignment '97), PRA97 (permission-role-assignment '97), and RRA97 (role-role assignment '97). URA97 was previously defined by Sandhu and Bhamidipati [SB97].

ARBAC97 incorporated URA97, built upon it to define PRA97 and some components of RRA97, and introduced additional concepts in developing RRA97.

URA97 is concerned with administration of the user-assignment relation UA which relates users to roles. Authorization to modify this relation is controlled by administrative roles. See Figure 2.1.

Director (DIR)

Project Lead (PL1) Project Lead 2 (PL2) Senior Security Officer (SSO)

Production Engineering 1 (PE2) Quality Engineeing 1 (QE1) Production Engineering 2 (PE2) Quality Engineering 2 (QE1)

Engineering 1 (E1) Engineering 2 (E2)

Department Security officer (DSO)

Engineering Department (ED)

Project Security Officer 1 (PSO1) Project Security Officer 2 (PSO2)

Employee (E)

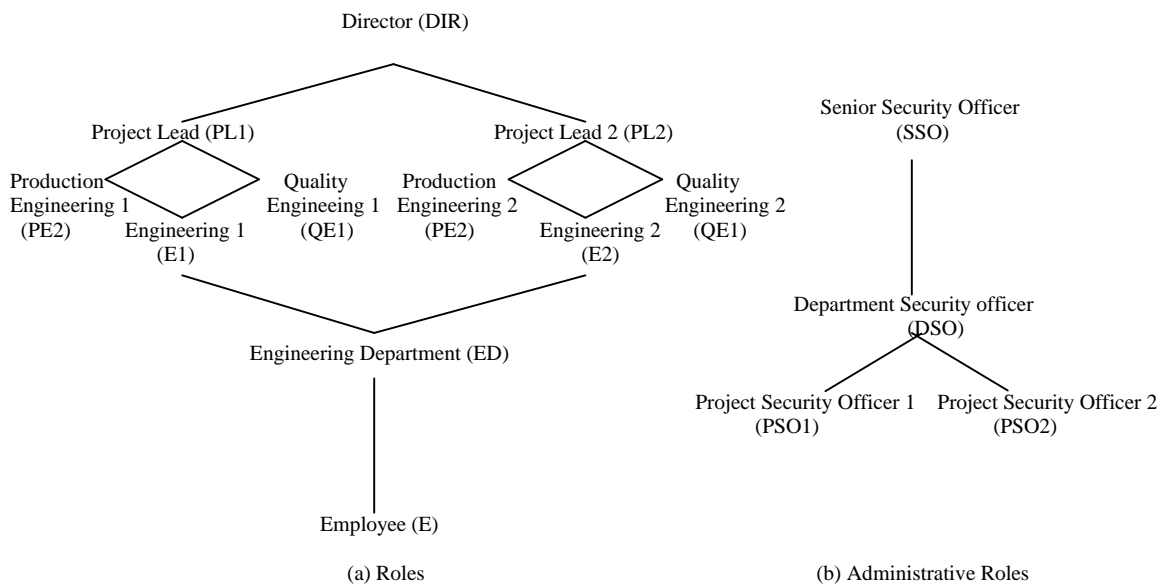(a) Roles (b) Administrative Roles

Figure 2.1. Example of Role and Administrative Role Hierarchies

Figure 2.1 shows that URA97 uses two hierarchies: one hierarchy represents regular roles in an organization and the other hierarchy is used to represent administrative roles. The administrative role hierarchy has the senior security officer (SSO) role at the top, two

project security officers' roles (PSO1 and PSO2), and a department security officer (DSO) role.

In URA97, members of the administrative roles hierarchy are authorized to modify membership in the regular roles hierarchy.

In URA97, user-role assignment and revocation are respectively authorized using the following relations: can-assign $\subseteq$ AR x CR x 2^R and can-revoke $\subseteq$ AR x 2^R.

The meaning of can-assign (x, y, Z) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can assign a user whose current membership, or non-membership, in regular roles satisfies the prerequisite condition y to be a member of a regular role on range Z^2. The meaning of can-revoke (x,Y) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can revoke membership of a user from any regular role y $\in$ Y.

Zhang, Ahn, and Chu [ZAC01] proposed a rule-based framework for role-based delegation including the RDM2000 model and rule-based specification language. One of the contributions of their work is that their model supported multi-step delegation. They proposed a rule-based declarative language to specify and enforce policies.

2.2 Work Indirectly Related to Delegation

Propagation of permissions can also be considered as delegation. A large number of papers have been published in this area. Some of the well-known models are the Access Matrix, HRU, TG, SPM, TAM, and ATAM.

The Access Matrix proposed by Lampson [Lam71] contains a particular set of rules to control the propagation of access rights. These rules basically give the owner of an object complete discretion regarding rights to that object.

The propagation model known as HRU [HRU76], formalized by Harrison, Ruzzo, and Ullman, could easily express complex policies for propagation of access rights. This model has broad expressive power. Unfortunately, HRU has weak safety properties. In general, safety is undecidable (i.e., the determination of whether or not a given subject can ever acquire access to a given object is not guaranteed). Safety is undecidable for most policies of practical interest even in the monotonic version of HRU. Monotonic models do not allow deletion of access privileges. As the ability to delete access privileges is an important requirement, monotonic models are too restrictive to be of much practical use. It appears that HRU does not have any useful special case for which safety is efficiently decidable. A number of protection models were developed in response to the negative results of HRU.

The take-grant model, which was introduced by Lipton and Snyder [LS77], was deliberately designed to be of limited expressive power so that it would not exhibit the undecidable safety of HRU. There is therefore a substantial gap in expressive power between take-grant and HRU.

Sandhu's Schematic Protection Model (SPM) [San97] was developed to fill the gap in expressive power between the take-grant and HRU while sustaining efficient safety analysis.   The key notion introduced in SPM is that of strong security types: every subject or object is created of a particular type, which thereafter does not change. SPM led to the development of the Extended Schematic Protection Model (ESPM) [AS92a] by Ammann and Sandhu. ESPM extends single parent certain operation in SPM to allow multiple parents for a child. ESPM is equivalent to monotonic HRU [AS92a] in terms of expressive power.   It still retains the positive safety results of SPM.

The typed access matrix (TAM) [San 92], which was also formalized by Sandhu, introduced strong typing to HRU (i.e., each subject or object is created to be of a particular type which therefore does not change). TAM has the same expressive power as HRU. In order to accommodate both the strong expressive power of HRU and the positive safety results of ESPM, Ammann and Sandhu proposed the Augmented Typed Access Matrix (ATAM), which was an extension of TAM. ATAM is the same as TAM with the additional ability to test for absence of access rights. Hence, ATAM can express all the policies that can be expressed by TAM.

Chapter 3

FRAMEWORK FOR ROLE-BASED DELEGATION MODELS

3.1   Introduction

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. Delegation in computer systems can take many forms: human to human, human to machine, machine to machine, and perhaps even machine to human. This dissertation focuses on the human to human form of delegation using roles. As we will see, there are many different ways in which role-based human-to-human delegation can occur. I develop 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, and using a systematic approach to reduce the large number of cases into a few useful cases which can be used to build practical delegation models.

Human to human delegation has not received much attention in the literature so far. In this chapter I propose a framework for building good cases that can be used for developing role-based delegation models. This is the first systematic attempt toward

addressing the problem of delegation between humans using roles. I emphasize that the delegation itself occurs within the computer system even though it is described as human to human.

My approach to develop the framework begins by identifying a number of characteristics related to delegation between humans, using these characteristics to create an exhaustive combination of possible delegation cases, and implementing a systematic approach to reduce the large number of possibilities to a few useful cases. These cases are used for formalizing aspects of delegation based on Role-Based Access Control Model (RBAC) [San96]. This work involves the investigation and formalization of role-based delegation models using nine different delegation characteristics, as defined below.

3.2   RBDM Framework

In this dissertation, the approach to develop a framework for role-based delegation models was an exploratory approach. It began with the identification of a number of characteristics related to delegation between humans. The identified characteristics comprise permanence, monotonicity, totality, administration, levels of delegation, multiple delegation, lateral agreements, cascading revocation, and grant-dependency revocation. I assert that this list is an exhaustive list of characteristics although I would consider any additional characteristics that others may propose. Trying to address every characteristic as mutually exclusive is a formidable task, and it can be very complicated.

Therefore, a systematic approach is used in order to reduce the large number of possible cases. These reduced cases, which can be useful in business today, can be used to build delegation models. The following section provides definitions and explanations of these characteristics.

3.2.1 Definitions of Characteristics

The following is a list of definitions of the characteristics that are related to delegation:

1. **Permanence (permanent/temporary):** Permanence in role-based delegation models refers to types of delegation in terms of their time duration. Permanent delegation refers to delegation wherein another user who is a member of another role permanently replaces the delegating user. In this type of delegation, once the delegating user delegates his role, he or she can no longer take it back (but perhaps can get it back through some other means such as a security officer, or a reverse delegation). Upon delegation, the user who received the delegation assumes the same full power as any of the original role members. On the other hand, temporary delegation refers to delegation that is limited by time. Once that time is expired, the delegation is no longer valid. For example, a professor who is a member of a role called 'Advising Committee' and is no longer able to serve in the committee can permanently delegate his membership in that role to another professor. In this case the delegated professor becomes a permanent replacement

to the delegating professor. Also, the delegating professor loses all of his permissions in that role and cannot get them back unless the security officer assigns him back to that role. On the other hand, that same delegating professor may elect to delegate his role temporarily to his secretary to perform a task on his behalf. In this case, the delegating professor will not lose his power in that role and can revoke that delegation as he wishes.

2. **Monotonicity (monotonic/non-monotonic):** Monotonicity refers to the state of the authorization that the delegating role member possesses after he or she delegates the role. A monotonic delegation means that upon delegation the delegating role member maintains the power of his or her role. This user continues to be able to perform the same operations he or she had before delegating his or her role (e.g., he or she can override any action taken by the delegate member).

On the other hand, with a non-monotonic delegation, upon delegation the delegating role member loses the power of the delegated role for the duration of the delegation. In this case, the delegating role member will not be able to operate inside the delegated role that he/she delegated. This type of delegation is not permanent, because, once the delegation is revoked or expired, the delegating role member will regain full power over that role. For example, under the same professor analogy used in the example above, if the delegating professor non-

monotonically delegates his role to his secretary, then as long as that delegation is active the professor will not be able to act in the delegated role. However, he is still responsible for the behavior of the delegated member in that role, which, in this case, is his secretary.

3. **Totality (total/partial):** Totality is another characteristic of delegation. This term refers to how completely the permissions assigned to that role are delegated. There are two options: total delegation and partial delegation. Total delegation means delegating all of the permissions that are assigned to the delegated role. On the other hand, partial delegation means that only subsets of the delegated role are delegated. For example, a director of a clothing factory may elect to delegate his or her marketing sub-role to the marketing manager or his or her production sub-role to the production manager. Partial delegation is much easier to address using role hierarchies. For example, a professor in a university who has teaching assistants, lab assistants, and a secretary may delegate only part of his roles to different people. (For example., he may delegate his teaching role to his teaching assistant, his research role to his lab assistant, and his administration role to his secretary in order to handle his e-mail.)

4. **Administration (self-acted/agent-acted):** The term administration will be used here to describe the actual administrator of the delegation. There are two administration types for delegation: self-acted delegation, wherein the delegating

role member administers the delegation herself; and agent-acted delegation, wherein the delegating role member nominates a third party to conduct the delegation on his or her behalf. Regarding the latter, there are situations in which it is more convenient to have a third party administering the delegation on behalf of the user (e.g., in the case in which the delegator is not available, or is unable to perform the delegation). Usually, the agent can delegate to anyone else, but cannot delegate to him or herself. For example, a professor may designate a third party (e.g. another professor who is a member of the same professor role) to administer any of the delegations on his behalf.

5.   **Levels of Delegation (single step/multi-step):**   This characteristic defines whether or not each delegation can be further delegated and how many times. Single step delegation does not allow the delegation to be further delegated. This means that the delegated member is not allowed to further delegate the role that has been delegated to him or her. Two-step or multi-step delegation allows the delegated member to further delegate his or her delegated role to a third user, and so on.

For example, in the example above, once a professor delegates his role to another professor, the second professor may further delegate that role to someone else. Changing the policy to a single step delegation, which can enforce the delegation to take place only once, can prevent this.

6.   **Multiple Delegation:** This type of delegation refers to the number of people to whom a delegating role member can delegate at any given time. Sometimes, a delegating role member needs to delegate his or her role to more than one person at the same time. For example, a professor may need to delegate his or her access right to a certain database to more then one of his or her research assistants. This type of delegation is more effective if the delegation is temporary. With permanent delegation, delegating a role to more than one person at the same time can create accountability problems and increase the number of people in the role.

7.   **Agreements (bilateral/unilateral):** The term 'agreement' is used here to address the delegation protocol between the delegator and the delegated members. It is of two types: bilateral agreement and unilateral agreement. A bilateral agreement is an agreement wherein delegation is accepted by both the delegating role member and the delegated member. The delegating member is agreeing to delegate the role, and the delegated member is agreeing to accept the role. A unilateral agreement, on the other hand, is a one-way decision. Only the delegating role member can decide to delegate the role. Once the delegating member identifies a role member to whom he or she delegates the role, the delegated member has to accept that responsibility. The security implication for this type of delegation is that people sometimes use it for denial of service. In systems in which everyone is assigned a certain quota of memory, someone can create files and delegate them

to someone else causing the second person to exceed the limit, which can lead to a denial of service.

In order for a professor to delegate his role to another professor (or to any of his assistants), the second has to first agree to accept that responsibility before the delegation can take place

8. **Revocation**: Revocation refers to the process by which a delegating user can take away the privileges that he or she delegated to another user who is a member of another role. There are some interesting issues involving revocation. Among these issues are those raised by cascading revocation and by grant-dependent revocation. The following is a brief description of these types of revocations:

8.1. **Cascading Revocation:** This type of revocation refers to the indirect revocation of membership as a result of the revocation of the membership of some other related roles (i.e., supporting role or sponsoring role). With the supporting role (the role to which the delegated user belonged prior to being a delegated member of the new role), if the delegated member loses his or her membership in his or her supporting role, then as a result he or she will lose his or her delegated membership in the new role. In the case of the sponsoring role (the role that is responsible for bringing in the

delegated member), if that role is revoked, then the membership that was delegated by that revoked role will also be revoked.

For example: suppose that Alice is an original member of role a, and Bob is an original member of role b. Now suppose that Alice delegates her role to Bob who delegates that same role to another user, Charlie. Then, revocation will have the following scenarios:

- If Alice revokes Bob's delegate membership in role a, then as a result, Charlie will also lose his delegate membership in role a.
- If Alice loses her membership in role a then both Bob and Charlie will lose their delegate memberships in role a.
- If Bob loses his original membership in his own role b, then he will also lose his delegate membership in role a, and consequently, Charlie will also lose his membership in role a.

8.2. **Grant-Dependency (grant-dependent/grant-independent):** Grant-dependency refers to the decision that has been made regarding who may revoke the membership of the delegated member in a role. In the case of grant-dependent delegation, only the delegator is allowed to revoke the membership of the delegated member. Grant-independent delegation, on the other hand, allows any member in the sponsoring role to revoke the

membership of the delegated member. This decision becomes important, for example, if the delegated member behaves badly: with grant-dependent delegation, it could take a long time (depending on the delegator) to revoke the offender's membership. With grant-independent delegation, that membership can be revoked quickly by any original member of that role (which may of course create some friction between the original members).

Using different combinations of the above characteristics gives us a very large number of possible modes in which to perform delegation. Therefore, I have pursued a systematic approach in order to reduce this large number of cases into a smaller number of useful ones.

The list of delegation characteristics mentioned above is not exclusive; however, it reflects the result of exploratory research that I did in the literature of this field and my analysis of this topic.

The following section explains the framework I used to identify the useful cases for developing role-based delegation modes.

3.2.2   Reduction Approach

I have adapted a systematic approach by which I can reduce the large number of possible cases into a few useful ones.

I first partitioned delegation based on its permanence (permanent or temporary delegation). I believe it is useful to develop delegation models that support the implementations of the permanent and temporary delegation policies.   I further partitioned both the permanent and temporary delegations.   The permanent delegation was partitioned based on its monotonicity, whereas the temporary delegation was partitioned based on its level of delegation. Finally, I partitioned single step delegation of the temporary delegation based on its monotonicity.   After the partitioning was done, I then added the rest of the characteristics (one at a time) to each node and tested for combinations that are useful in business today and that can be used in developing delegation models. Figure 3.1 shows the combinations that appear useful in practice.

On the permanent side, I could make the claim that there is one clear path that can be followed to develop a delegation model. That path includes the following characteristics: permanent,   non-monotonic,   self-acted,   total   and   multi-step   delegation.   Other characteristics do not have much effect on the model and were therefore ignored.

On the temporary side, however, there are a number of possibilities; therefore, I have to make some simplification in order to identify useful combinations. The simplification is to eliminate multi-step delegation, given that dealing with multi-step delegation is a very complicated issue.

Delegation

Permanent

Temporary

Non-monotonic

Monotonic

Single step

Multi-step

Self-acted
Total
 Multi-step

Monotonic

Non- Monotonic

(eliminated)

Self

Agent

Total/Partial
G. D. revocation
 Cascading R.
Multi-delegation.
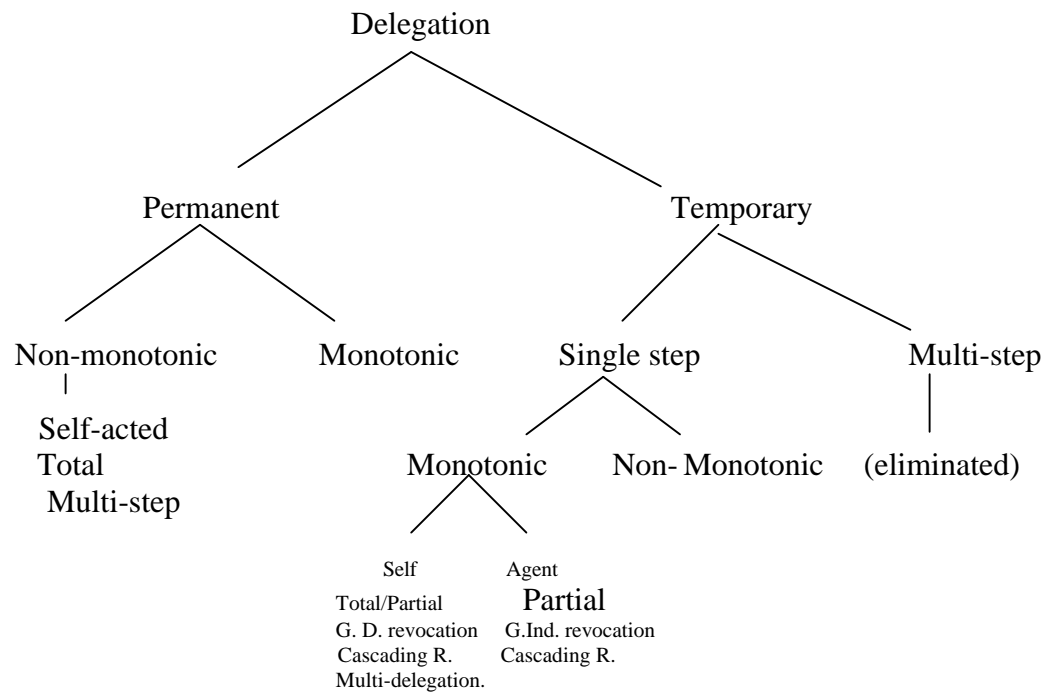
Partial

G.Ind. revocation
Cascading R.

Figure 3.1. Framework for Role-Based Delegation Models

The following sections explain the reduction process and provide rationale for selecting these combinations of characteristics. I start by examining the permanent side followed by the temporary side.

3.2.2.1 Permanent Delegation

Non-monotonicity

Permanent-monotonic delegation does not appear to be very desirable. If a user in a role elected to delegate his or her role permanently to another user who is not a member of that role, the delegated member becomes his or her replacement in that role. In such case there is no need for the delegating role member to maintain his or her power in that role. Doing so would increase the number of users in the delegated role.

Permanent non-monotonic delegation, on the other hand, seems to be more useful. Once a delegating user permanently delegates his or her role to another user and loses his or her power in that role, then he or she is no longer responsible for the behavior of the newly delegated member. Non-monotonic delegation allows the new role member to act independently and as if he or she is an original member of that role. For example, a member of an advising committee (say, Alice) decides that she can no longer serve in the committee and is willing to permanently delegate her role to someone else who is not a member of this committee (say, Bob). In this scenario, it makes sense that the delegation to Bob by Alice be non-monotonic. Alice loses all of her power in that role. In fact, to preserve the security of committee role, the delegation must be non-monotonic. This is because Alice relinquishes her membership of that role.

Totality

When the delegation is permanent and non-monotonic, partial delegation is not desirable because delegating only a subset of the role does not allow the delegated member to carry out the full task of the delegating member. For example, suppose that Alice (who is an original member of Role a) permanently, and non-monotonically delegates only a subset of her role to Bob (who is a member of Role b). In this case, Bob will not be able to fully carry out the task of Alice (e.g., if the task of reviewing student records was not delegated, then Bob will not be able to provide efficient advice to the students). Moreover, neither Alice nor Bob will end up with full power in that role after the delegation. This is because upon delegation, Alice will not retain any power in the delegated role.

Administration

With permanent, non-monotonic, total delegation, having an agent doing the delegation does not seem to be very appealing; I think it is more sensible if the delegating member him or herself administers the delegation. Delegation of this kind is usually planned in advance, and there is no extreme need to rush it. Thus, there is no need for the delegating member to nominate someone else to do the delegation on his or her behalf

Levels of Delegation

If the delegation is permanent and non-monotonic, then it is up to the delegated member to decide whether or not he or she wants to delegate his or her role further. Therefore, from a modeling perspective, the level of delegation is irrelevant.

Agreement

With this combination of characteristics, bi-lateral agreement between the delegating member and the delegated member is essential, because if the delegated member does not accept the responsibility of the delegating role, then the purpose of the delegation is defeated and there is no need for it to take place.

Revocation

With permanent and non-monotonic delegation, revocation becomes irrelevant. Once a user becomes a permanent delegated member in a different role, then he or she assumes the full power in that role, and only the security officer can revoke his or her membership in that role.

In summary, with permanent delegation, it is desirable to let the model be simple, yet tight (no agent should be involved, and there should be no partial delegation). Therefore,

it makes more sense to choose the following combination: permanent, non-monotonic, self-acted, total delegation. Other characteristics are not relevant and therefore are ignored.

3.2.2.2 Temporary Delegation

With temporary delegation, identifying specific paths to develop role-based delegation models is not as obvious as it was with permanent delegation. In the case of temporary delegation, there are many possibilities, and the process of selecting paths that make the most sense for developing delegation models is quite complicated. For example, if I test for useful combinations using the administration characteristic in combination with the temporary delegation characteristic, I cannot make a clear determination as to which characteristic makes more sense (i.e., with the temporary delegation I cannot determine whether the self-acted or the agent-acted delegation makes more sense). Therefore, to scope the path in the case of the temporary delegation, I assume:

- Everything is bilateral.
- Only single step delegation is addressed.
- Only monotonic delegation is addressed.

The following section discusses the testing process related to temporary delegation. For simplicity, I used the administration characteristic to further partition these characteristics.

3.2.2.2.1 Self-Acted Delegation

In combination with temporary and self-acted delegation, monotonic delegation has two advantages. First, it prevents the delegated member from having exclusive power over the role, especially because the delegator is still responsible for that role. Monotonic delegation allows the delegator to possibly review and correct any wrong action that might be taken by the delegated member. Second, the delegator can take over the task of the delegated member in case the delegated member suddenly loses his or her membership.

With these characteristics it seems useful to consider both partial and total delegation. The delegator should be able to delegate all of his or her role or just part of it. In both cases, this will not conflict with the permanent, self-acted, monotonic delegation.

As we will see in Chapter 4, total delegation is handled in my delegation model-flat roles (RBDM-FR), and partial delegation in my delegation model-hierarchical roles (RBDM-HR).

Most published research in the area of the Discretionary Access Control (DAC) suggests that multi-step delegation is a very complicated issue and that dealing with it can create many problems; therefore, with temporary delegation, multi-step delegation is eliminated from my consideration. I consider only single step delegation.

With the delegation being temporary, there are situations where is it more useful to have the delegator himself delegate his role to different other users (i.e., where he only trusts specific individuals to do a certain task). Therefore, it is useful to consider multiple delegation in this case.

With this combination of characteristics, it is more appealing to consider both types of revocation: grant-dependent and grant-independent revocation. It is also more appealing to consider the cascading revocation issues with this combination of delegation characteristics.

Therefore, with the delegator as the administrator of the temporary delegation, I select the path temporary, self-acted, single step, monotonic, partial/total delegation, grant-dependent/grant-independent, and cascading revocation.

3.3   Summary of RBDM Framework

I have identified some characteristics related to delegation between humans in computer systems. I then used a systematic approach to reduce the large number of possible cases to a few cases that can be useful in business today. I partitioned the characteristics into permanent and temporary delegations. On the permanent side it was straightforward to determine a path that can lead to the development of a delegation model. On the temporary side, there were many possibilities that led us to make some simplifications. These simplifications included the elimination of the multi-step and the non-monotonic delegations.

Chapter 4

ROLE-BASED DELEGATION MODEL WITH SELF-ACTED TEMPORARY DELEGATION (TRBDM)

4.1   Introduction

In this chapter and the next, I derived some models to illustrate some good access control policies using the delegation cases I arrived at using the framework for role-based delegation model discussed in the previous chapter.

This chapter addresses delegation between roles in a model where the delegation is temporary and self-acted. My work is cast within the framework of the well-known RBAC96 model [San96]. I first use the simplest form of the RBAC model, RBAC0, to address delegation based on the flat relation between roles, with no hierarchical role relation. Next, I extend this model to include hierarchical roles. To accomplish this, I have developed two different models. The first model that I have derived in this chapter, which I call RBDM0, is based on flat relations between the roles involved. The second model, which I call RBDM1, is an extension of RBDM0. It addresses delegation between roles in hierarchical role relations.

As summarized in Figure 4.1, a user is a human being, a role is a job function, and permission is an approval of access to some objects or a privilege to carry out a particular task. The management of permissions and roles is greatly simplified by associating permissions with roles and assigning the users to these roles. In this way the users acquire the associated permissions. Roles are created for various job functions in an organization. The permissions required to carry out the jobs are associated with the roles. New permissions can be granted to roles as new applications and systems are incorporated. Likewise, unnecessary permissions can be revoked from the roles. Users are assigned to roles depending on their responsibilities and qualifications, and users can be reassigned from one role to another. (The session concept of RBAC96 is not used in my work and is hence omitted here.)
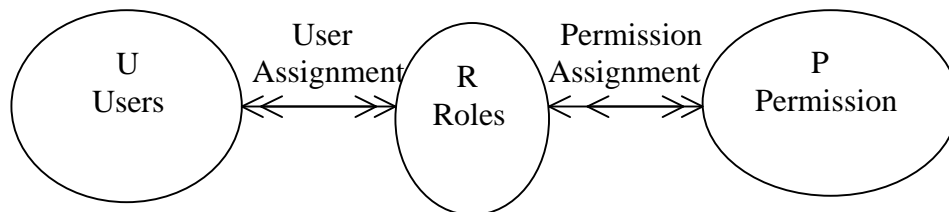

Figure 4.1. Simplified Version of RBAC96

This chapter contains two subsections. The first subsection (4.2) addresses the delegation between roles that have a flat relation between them. The second subsection (4.3) of this chapter addresses delegation between roles that have a hierarchical relation between them.

4.2   Role-Based Delegation Model / Flat Roles (RBDM0)

This model, which I call RBDM0, is the simplest form of the RBDM models and is based on RBAC0 of the RBAC96 family. This means that the delegation addressed in this section is between users in flat roles (no inheritance of permissions between roles is involved), and no constraints are considered. First, I provide some assumptions and basic elements that I will use throughout this section.

4.2.1   Assumptions and Basic Elements

Delegation between members in the same role is not allowed because it is meaningless. This assumption is very basic and it will not be relaxed throughout the dissertation.

The delegation addressed in this model is a one step delegation.   This means that the delegated role cannot be further delegated. Hence, only the original members can delegate. I keep this assumption in my original model, but, as part of future work, I recommend that this assumption be relaxed to allow some multi-step delegation.

The delegation is total. Each user in a delegating role delegates the total package of permissions embodied in that role or does not delegate at all. This assumption will also

be relaxed later when I extend my model to include partial delegation in context of hierarchical roles.

Each delegating role r has two types of members:

1. Original members, Users_O(r), are the members who were originally assigned to the role by the system administrator.

2. Delegated members, Users_D(r), are the members who are assigned to the role by other original members (who are assigned by delegation).

To simplify revocation, I have assumed in my basic model that any original member in a role can revoke the delegation to any delegate member in that role. In other words, revocation is not related to who performed the delegation. As we will see, revocation is one area that presents many different policy choices, some of which will be explored in the hierarchical model (RBDM1).

I have assumed each unit of delegation has a time element associated with it called duration (T). The duration of each delegation is under the control of the delegating user. Once the assigned time for the delegation expires, the delegation is automatically revoked. Members in the delegating role can also exercise revocation of delegation even if the duration of delegation is still valid.

The following definitions formalize the above discussion. For convenience we first repeat the definitions of RBAC96 without flat roles.

Definition 4.1: The original RBAC96 model (with flat roles) is comprised of the following components:

- U and R and P are sets of Users, Roles, and Permissions, respectively.

- $UA \subseteq U \times R$ is a Many to Many, User to Role assignment relation

- $PA \subseteq P \times R$ is a Many to Many, Permission to Role assignment relation

- Users: $R \rightarrow 2^U$ is a function derived from UA mapping each role r to a set of users where Users(r)   = {U | (U, r)$\in$UA}

- Permissions: $R \rightarrow 2^P$ is a function derived from PA mapping each role to a set of permissions where Permissions (p) = {P | (P, r) $\in$ PA}

The following additions give us RBDM0.

Definition 4.2: The RBDM0 model adds the following components to the previously defined RBAC96 model:

- $UAO \subseteq U \times R$ is a Many to Many, Original Member to Role assignment relation

- $UAD \subseteq U \times R$ is a Many to Many, Delegate Member to Role assignment relation

- $UA = UAO \cup UAD$

- $UAO \cap UAD = \varnothing$ Original members and delegate members in the same role are disjoint

- $Users\_O(r) = \{U \mid (U, r) \in UAO\}$

- $Users\_D(r) = \{U \mid (U, r) \in UAD\}$

- All members $Users\_O(r) \cup Users\_D(r)$ in a role receive all of the permissions assigned to that role

- Note that $Users\_O(r) \cap Users\_D(r) = \varnothing$ because $UAO \cap UAD = \varnothing$

- T is a set of durations

- Delegate roles: $UAD \rightarrow T$ is a function mapping each delegation to a single duration

To complete RBDM0 it remains to define the can-delegate relation below.

4.2.2 Delegation in RBDM0

The concept of role delegation in RBDM0 has some similarity to that of the user-role assignment component (URA97) of the ARBAC97 [San, Bha, and Mun 97].

URA97 is concerned with administration of the user-assignment relation UA which relates users to roles. Authorization to modify this relation is controlled by administrative roles.

In RBDM0, the delegation of a role of a user in a delegating role to another user in a different role is actually making the delegated user a member of the delegated role. This function is handled by the delegating user himself. This function is different from the user-role assignment of ARBAC97 in that in ARBAC97 there are two aspects to this decentralized user-role assignment in URA97. First, roles whose membership can be modified by administrative role must first be specified. Secondly, the population of users eligible for membership must also be specified. Also, assignment of the users to the administrative roles is done through the security officer.

In RBDM0, user-to-user role delegation is totally discretionary. The delegating role member can delegate to whomever he or she wishes. This function is a widely decentralized task and is taken care of by the users themselves without continuous involvement from the security officer.

User-user delegation is authorized in RBDM0 using the following relation:

Definition 4.3: RBDM0 controls user-user delegation by means of the relation can-delegate $\subseteq R \times R$.

Can-delegate is irreflexive. This means that a user in a role cannot delegate his membership to another user in the same role, since this is meaningless.

The meaning of (a, b) ∈ can-delegate is that a user (say, Alice) who is an original member of role a can delegate her role membership to any another user (say, Bob) who is an original member of another role b. For example, if Alice ∈ User_O(a) and Bob ∈ User_O(b), then Alice can delegate to Bob, so thereby Bob∈ User_O(a).

4.2.3   Revocation in RBDM0

So far I have described how users in a delegating role can delegate their permissions to other users in other roles, and how I can control this process using the can-delegate relation. However, as often happens in real life, we may want to revoke rights. In the examples described above, when the department chairperson goes away, one or more other professors will be delegated the chairperson's permissions. Subsequently, when the department chairperson returns, the delegated permissions need to be removed from the delegated professors. In this section I look at possible ways in which a user in a delegating role can change his/her mind and revoke the permission that he/she delegated. I have also considered under which conditions it is not possible to revoke a previous decision and the issues that might arise as a result of revocation.

4.2.3.1   Types of Revocations

RBDM0 deals with the issue of revocation in two ways: (1) by using timeouts and (2) by allowing any original member of the delegating role to revoke the membership of any delegate member in that role. The following two subsections describe both approaches and discuss the pros and cons of each approach.

1. Revocation Using Time Out

In using this approach, I attach a time clock to every assigned delegation so that when the assigned time expires, the delegation also expires.

The following is an explanation of the concept of delegation duration:

Duration (D) is an important feature in RBDM0. It is an interval of time (it has a starting and ending time) assigned to each delegation. It changes in a decreasing fashion throughout the life of delegation. Suppose that the delegation has a starting date, starting time (birth of delegation) and ending date and time. Then when the value of the ending date and time is reached, the delegation will no longer be valid (expires automatically).

Duration is used in this model as a tool to automatically revoke a membership in the absence of the delegator (or the security officer). For example, if Alice $\in$ User_O(a) and

Bob $\in$ User_O(b), then Alice can delegate her membership to Bob for a specified duration (i.e. τ = <10.2,1300,24>), so, therefore, (Bob, a) $\in$ UAD starting on Oct. 2, at: 1300 for 24 hours. When the 24 hours expire then the delegation expires and Bob will be automatically revoked from a.

This approach has some advantages and some disadvantages.

Using timeouts has the following advantages:

1. Timeout revocation is a simple self-triggering process that ensures the revocation of delegate membership automatically.

2. In attaching a timeout to the delegation I no longer have to worry about tracking the sponsoring roles.

Using timeouts has the following disadvantages:

1. Using timeouts by themselves is not enough to ensure security.

2. If there is no other tracking mechanism, delegate members can behave in a bad manner during the duration of the time set, which can cause great harm to the system before revocation takes place by time out.

3. When employing this approach, we have to choose the time carefully because we might over-set or under-set the time for delegate members.

2. Revocation by Role Members

Any original member in a delegating role can revoke the membership of any delegated member in that role.

- This gives the power to the original members to protect the role from the temporary delegate members.
- It raises the possibility of conflicts between the original members if someone else other than the sponsoring original member revokes the delegate member.
- If the delegate member behaves badly, any original member can revoke him immediately, which will minimize the damage even before the time out.

In RBDM0, there is no need to define a can-revoke relation. This is because a can-revoke relation requires identifying the different roles involved, but in this case there is only one role relative to the revocation process. Both, the delegating and the delegated members belong to the same role.

4.2.4   Summary of RBDM0

To summarize, the RBDM0 model has the basic elements given in Definition 4.2.1 and authorizes delegation using the can-delegate relation defined in Definition 4.3. Moreover, the model deals with the issue of revocation using the notions explained in Section 4.2.3.

4.3   Role-Based Delegation Model/ Hierarchical Roles (RBDM1)

This model is an extension of RBDM0, which I described in the previous section. This section describes the ways in which RBDM0 is extended to address more complicated issues that arise with hierarchical roles.

Hierarchies are natural means for structuring roles to reflect an organization's lines of authority and responsibility (figure 4.2). By convention, more powerful (senior) roles are shown toward the top of these Hess diagrams, and less powerful (junior) roles toward the bottom. In figure 4.2a, the junior-most role is that of the health-care provider. The physician role is senior health-provider and thereby inherits all permissions from health-care provider. The physician role can have permissions besides those it inherited. Permission inheritance is transitive. So, for example, in Figure 4.2a, the primary-care physician role inherits permissions from both the physician and health-care provider roles. The primary-care physician and the specialist physician both inherit permissions from the physician role, but each will have different permissions directly assigned to it. Figure 4.2b illustrates multiple inheritances of permissions, where the project supervisor role inherits from both the test engineer and the programmer role.

Mathematically, these hierarchies are partial order. A partial order is a reflexive, transitive, and anti-symmetric relation. Inheritance is reflexive because a role inherits its

own permissions, transitivity is a natural requirement in this context, and anti-symmetry

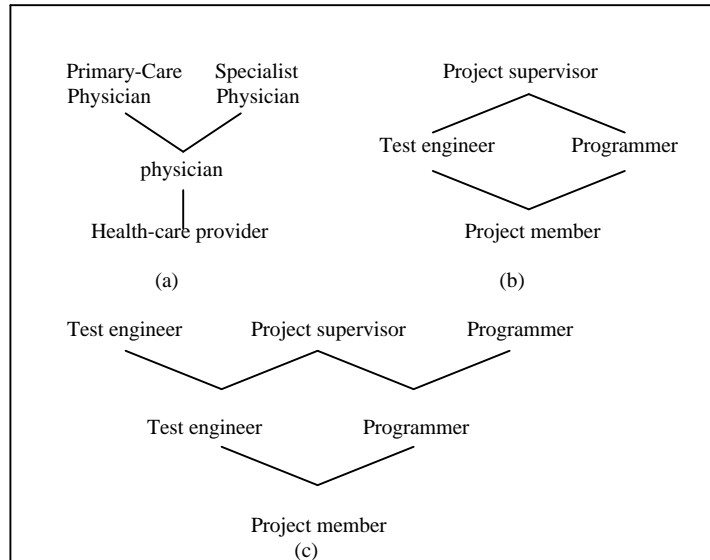rules out roles that inherit from one another and would therefore be redundant.

Figure 4.2. Example of Role Hierarchy

When we extend RBDM0 model to capture the role-to-role delegation using hierarchical

roles, we add more complexity to my flat model. Here, we have to deal with different

kinds of delegations, some of which are not very useful, and some which carry more risk

than others.

To appreciate the reason behind doing delegation in hierarchical roles, let us consider a

typical example from the office context. Suppose that we have a department whose

manager (DM) has access to view and modify the overall departmental portfolio (DP).

Now, let us suppose that the department has several projects, each of which has an individual portfolio (Dpi). A project manager (PM) can view or modify the project's portfolio if and only if the departmental manger (DM) has delegated the appropriate right to it. In this case, the project manager (PMi) is acting on behalf of the departmental manager. On some occasions, the departmental manager may only wish to give the project manager the right to view another project's budget without allowing him to perform any modifications. So, a user in a role may delegate all or only a subset of his role to another user who belongs to another role. Furthermore, a department manager may delegate the membership of one project manager to a project member, or to another project manager. Also, a project manager may delegate his delegated rights over the budget to a project member (this is known as two step delegation and is not allowed in my model). These types of situations are common in many business organizations.

For each object involved in a delegation, there are certain requirements that have to be met. The originator, or delegator, may wish to give only a part of its overall rights, or even just a single right. Furthermore, he may only want to grant these rights for a limited duration. Also he should be able to identify each of his delegations so that he may at some stage attempt to revoke one or all of these delegations.

The needs above can be justified by explaining delegation as a particular mechanism for collaborative working. Suppose a group of employees need to work together. In delegation, the members of the group do not work in tandem; their rights are used by

delegates of the group without their participation. This results in a need for trust between members. This trust can be limited in scope by limiting the rights contributed by delegator to delegate.

The most familiar form of collaborative working is hierarchical in nature, as shown in the office example above. In such hierarchical cooperation, the superior might not take part in the details of a task, but he or she is the instigator of the task, and participates through granting authority, and even talking to users who are his junior.

In this section, I have formally defined a role-based delegation model based on hierarchical relationship between the roles involved. I have also identified the different semantics that impact the can-delegate relation, analyzed these semantics to determine which ones I consider as more appropriate in business today, (thus allowed in my model) and provided a justification to why those selections are made.

The rest of this chapter is organized as following: Section 4.3.1 provides assumptions and basic elements that are specific to the role-based delegation models in hierarchical roles. Section 4.3.2 discusses delegation in RBDM1, and analyzes the different semantics of delegation in hierarchical roles. This is addressed in sub-section 4.3.2.1. Section 4.4 addresses revocation of delegation within RBDM1. Finally, Section 4.5 provides a summary of the entire TRBDM model.

4.3.1 Assumptions and Basic Elements

In addition to the elements discussed in RBDM0 (delegation in flat) this section adds the following assumptions and basic elements that apply specifically to the delegation model using hierarchical roles:

- Delegation can only be either downward or cross. Upward is useless because senior roles inherit all the permissions of their junior roles.

- Downward delegation means that a user who is an original member of a role delegates his role to other users who are original members of roles that are junior to the delegation role.

- Cross delegation means that the delegation takes place between users who are members of incomparable roles. For example, a manager in the sales department can delegate his role membership to an auditor from the auditing department in order to do auditing on the sales department.

Unlike RBDM0, in RBDM1 partial downward delegation is allowed because members of senior roles can delegate only subsets of their permissions (only enough to accomplish the task). Partial cross delegation is also allowed.

Original members of senior roles are also original members of the roles that are junior to their roles, and delegate members of senior roles are also delegate members of the roles

that are junior to their roles. However, this type of membership is considered an implicit membership.

The addition of role hierarchy to RBDM0 introduces a new notion for a user membership in a role (explicit and implicit memberships). Explicit role membership grants a user the authority to use the permissions of that role because of his/her direct membership to that role. Implicit role membership, on the other hand, grants a user the authority to use the permissions of that role because of the user's membership in a role that is senior to that role.

Combining the two new types of role memberships with the earlier two types (original memberships and delegate memberships) produces four different combinations of user memberships in a role at any given moment. These combinations are: original/explicit, original /implicit, delegate/explicit, and delegate/implicit. These combinations will have a major impact on the semantics of the can-delegate relation in this model.

Revocation issues become more complicated when we deal with hierarchical roles. This is because of the involvement of many different roles and their hierarchical relationships.

To show the natural progression from RBDM0 to RBDM1, the following definitions of RBAC96 and RBDM0 are repeated here for convenience:

Definition 4.4. The following is a list of the original components of the RBAC96 model:
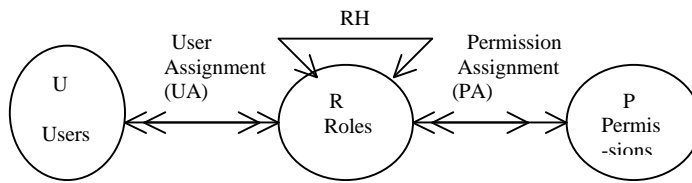

Figure 4. 3. Simplified Version of RBAC96

- U and R and P are sets of users, roles, and permissions respectively

- $UA \subseteq U \times R$ is a many to many, User to Role assignment relation

- $PA \subseteq P \times R$ is a many to many, Permission to Role assignment relation

- $RH \subseteq R \times R$ is a partially ordered role hierarchy (this can be written as $\geq$ in infix notation)

- Users(r) = $\{U| (U,r) \in UA\}$

- User: $R \rightarrow 2^U$ is a function mapping each role r to a set of users

- Permission: $R \rightarrow 2^P$ is a function mapping each role to a set of permissions

- Also, the less familiar symbol $||$ is used to denote non-comparability: we write $x||y$ if $x \neg \leq y$ and $y \neg \leq x$

Definition 4.5 RBDM0 adds the following components to the definition of RBAC96:
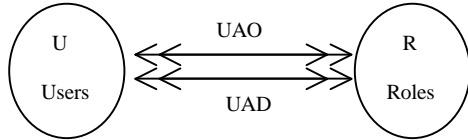

Figure 4.4. RBDM0

- $UAO \subseteq U \times R$ is a many to many, Original Member to Role assignment relation

- $UAD \subseteq U \times R$ is a many to many, Delegate Member to Role assignment relation

- $UA = UAO \cup UAD$

- $UAO \cap UAD = \varnothing$ original members and delegate members in the same role are disjoint

- Users_O $(r) = \{U| (U,r) \in UAO\}$

- Users_D $(r) = \{U|(U,r) \in UAD\}$

- Users$(r) = $ Users_O$(r) \cup$ Users_D(r)

- Users_O $(r) \cap$ Users_D $(r) = \varnothing$

- All members, Users_O $(r) \cup$ Users_O (r), in a role receive all of the permissions assigned to that role

Definition 4.6 The following is a derived definition of RBDM1:

The definition of RBDM1 is the same as RBDM0, with the following elements added (see figure 4.5):

- RH $\subseteq$ R $\times$ R is a partial order role hierarchy (this can be written as $\geq$ in infix
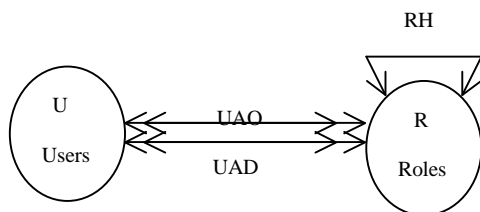
  notation).

Figure 4.5. RBDM1

4.3.2 Delegation in RBDM1

In RBDM1 my goal is to define a model by extending the RBDM0 model in order to capture the notion of delegation in the case of hierarchical roles and to show how the model handles the impact of the changes to the user-role assignment.

In RBDM1, authorization of delegation depends on the semantics of the can-delegate relation. These semantics become specially complicated when the membership statuses of the delegating and the delegated roles vary from one situation to another. For example, the delegation by an original explicit delegating role to an original implicit delegated role will carry a different meaning than a delegation by an original implicit role to an original explicit role, and so on.

In this section, I have addressed how the semantics of delegation in RBDM1 impact the can-delegate relation. I have listed a number of possible semantics for the can-delegate

relation in RBDM1, analyzed these semantics and identified the ones that make more sense for business today(thus allowed by my model) and I have justified my selections by giving some examples. Furthermore, in this section, I have addressed how revocation is handled under the new conditions.

The addition of role hierarchy to RBDM0 introduces a new notion for a user membership in a role (explicit and implicit memberships). Explicit role membership grants a user the authority to use the permissions of that role because of his/her direct membership to that role. Implicit role membership, on the other hand, grants a user the authority to use the permissions of that role because of the user's membership of a role that is senior to that role.

Definition 4.7 The user-role assignment is authorized in RBDM1 by the following relation: can-delegate $\subseteq R \times R$

In RBDM1, expressing and enforcing the delegation between users is done through the different semantics of the can-delegate relation. The following section introduces and explains the semantics used by this model to enforce the delegation between users that belong to different roles.

4.3.2.1 Semantics of Delegation in RBDM1

The semantics of the delegation relation become especially complicated when the relation between the roles involved is hierarchical. This is because along with the hierarchical relation comes an additional type of role memberships (explicit, implicit), which makes the meaning of the can-delegation dependent on the membership status of each of the delegating and the delegated roles in any given situation.

In this section, I have listed and analyzed the different semantics that impact delegation in RBDM1 and explained the approach my model takes towards allowing the appropriate semantics of delegation.

Figure 4.6 depicts organizational role hierarchy and users' role memberships. To illustrate the different semantics of delegation in RBDM1, I use this example in the rest of this chapter.
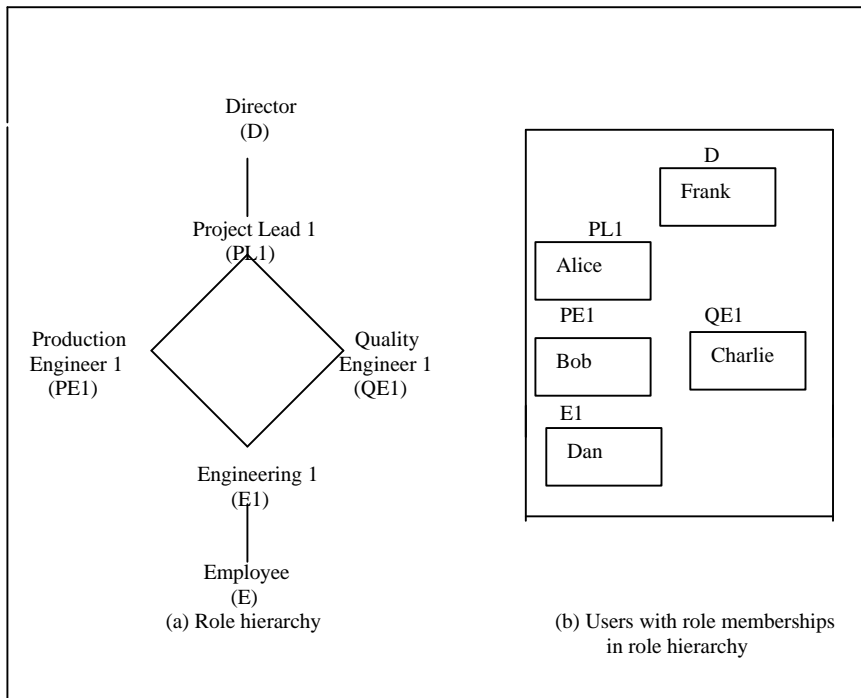
Figure 4.6. An Example of Organizational Role Hierarchy and Its Users

The following is a list of the semantics that control the authorization of delegation in RBDM1. The first three constraints are general constraints, and the fourth is a set of semantics that result from the different membership status in the delegating and the delegated roles at any given time.

1) $(x, y) \in$ can-delegate means that original members, explicit or implicit, of x can make an original member, explicit or implicit, of y an explicit delegate member of any other role junior or equal to x.

2) For x >y $\Rightarrow$ (y, x)$\notin$ can-delegate

   This means that a member of a role cannot delegate his role membership to another user who is a member of another role senior to his role. For example, in Figure 4.6, Alice who is a member of (PL1) cannot delegate PL1 to Frank who is a member of the role director, because by definition, Frank inherits the permission of role PL1.

   This constraint is very useful, because it prevents the delegation from being upward.

3) (x, y), (y, x) $\in$ can-delegate $\rightarrow$ x $||$ y

   (x, y), (y, x) $\in$ can-delegate means that users that belong to different roles can delegate to one another only if the roles to which they belong are non-comparable.

   This constraint is also useful because in some cases, in the office context, there is a need for a manager from one department to assume the responsibilities of the manager of another department and vice versa.

   For example, Bob who is a member of PE1 can delegate his role to Charlie who is a member of QE1 and vice versa.

4) The following sets of semantics are based on the statuses of both the delegating role and the delegated role (explicit/implicit) at the time of delegation.

For the sake of illustration I have used Table 4.1, in conjunction with Figure 4.6, to describe the derived semantics of the can-delegate relation. I have used all possible combinations of the delegating role/delegated role memberships.

As the case in RBAC96 and RBDM0, in RBDM1, delegating role members and delegated role members are assumed to be original members. Moreover, throughout this discussion, I have assumed all the members shown in Figure 4.6 to be original-explicit members.

I have used OE to denote original explicit members and OI to denote original implicit members. Hence the four possibilities are (OE, OE), (OE, OI), (OI, OE), and (OI, OI), where the first item of each tuple represents the delegating role member and the second represents the delegated role member.

The table below lists all different combinations that resulted form the above conditions.

Table 4.1. Examples of Authorization Functions

	Status of the role memberships		Given that (PL1, E1) **Î** can-delegate
	Delegating role	Delegated role	Semantics of can-delegate relations
RBDM0 (Flat roles)	OE	OE	Alice can delegate PL1 to Dan, and Dan can delegate to Alice
RBDM1 (Hierarchic al roles)	OE	OE	Alice can delegate PL1 to Dan Alice can delegate PE1 to Dan Alice can delegate QE1 to Dan Alice cannot-delegate PL1 to Bob Alice cannot-delegate PL1 to Charlie
	OE	OI	Alice can delegate PL1 to Dan Alice can delegate PL1 to Bob Alice can delegate PL1 Charlie Alice can delegate PE1 to Charlie Alice can delegate QE1 to Bob
	OI	OE	Frank can delegate PL1 to Dan Frank can delegate PE1 to Dan Frank can delegate QE1 to Dan Frank cannot-delegate PL1 to Bob Frank cannot-delegate PL1 to Charlie
	OI	OI	Frank can delegate PL1 to Dan Frank can delegate PL1 to Bob Frank can delegate PL1 Charlie Frank can delegate PE1 to Charlie Frank can delegate QE1 to Bob

The table above shows that in RBDM1 the meaning of the can-delegate relation changes depending on the explicit/implicit status of the (delegating and the delegated) roles involved in the delegation process.

With the assumption that (PL1, E1) $\in$ can-delegate, the following authorizations have been derived:

1. In RBDM0, where the relation between roles is flat, the can-delegate relation has very clear meaning: both the delegating and the delegated roles are original/explicit. Therefore, the can-delegate relation has one meaning: (PL1, E1) ∈ can-delegate. This means that any member of PL1 can delegate to any member of E1, and vice versa.

2. In RBDM1, the can-delegate relation has different meaning depending on the status of the delegating and delegated roles.

In the first scenario, where both the delegating and delegated roles are original explicit (OE, OE), (PL1, E1) ∈ can-delegate means that Alice can delegate PL1 to Dan, Alice can delegate PE1 to Dan, Alice can delegate QE1 to Dan. This is because of Alice's implicit membership in both PE1 and QE1. This also means that Alice cannot delegate PL1 to Bob, and Alice cannot delegate PL1 to Charlie. This is because both Bob and Charlie are explicit members in their respective roles, which means that they are also implicit members in E1.

This of course creates an anomaly, because Bob and Charlie are both senior to Dan, and it does not make much sense for Alice to be able to delegate PL1 to Dan and not to Bob and not to Charlie.

In the second scenario, where the delegating role is original/explicit and the delegated role is original/implicit (OE, OI), our table shows that because Dan is an implicit member

of E1, he is also an explicit member of PE1 and an explicit member of QE1. This means that, in addition to being able to delegate PL1 to Dan, Alice can delegate PL1 to Bob, and Alice can delegate PL1 to Charlie. This also means that Alice can delegate PE1 to Charlie, and Alice can delegate QE1 to Bob.

In the third scenario, where the delegating role is original/ implicit and the delegated role is original/ explicit (OI, OE), our table shows that now Frank can delegate PL1 to Dan, Frank can delegate PE1 to Dan, and Frank can delegate QE1 to Dan. It also shows that Frank cannot-delegate PL1 to Bob and cannot-delegate PL1 to Charlie

In the last scenario, where both the delegating role and the delegated role are original/ implicit (OI, OI), our table shows that Frank can delegate PL1 to Dan, Frank can delegate PL1 to Bob, Frank can delegate PL1 to Charlie, Frank can delegate PE1 to Charlie, and Frank can delegate QE1 to Bob. This is not desirable because it prevents any explicit members from delegating.

In conclusion, in this model I have chosen the most liberal approach of authorizing delegation between users in different roles. This means that my model allows all semantics of the can-delegate relation. This is motivated by the fact that allowing one semantic or the other will produce anomalies. For example, allowing only (OE, OE) means that Alice will not be able to delegate PL1 to Bob, or to Charlie. However, Alice is allowed to delegate the same role to Dan, which is a less powerful role than that of Bob

and of Charlie. Also, allowing only (OE, OE) will prevent Frank from delegating PL1 to Dan. This is not desirable because Frank is the most senior role, and thus inherits permission of all other junior roles. Hence, Frank should be allowed to delegate PL1 to anywhere Alice can.

Finally, allowing only (OI, OI) to delegate is not desirable because allowing the implicit membership to delegate and not the explicit memberships. This puts more trust on the memberships gained via inheritance than on the ones that were originally assigned by the security officer.

The above semantics of delegation are a result of having a non-empty hierarchy. If the hierarchy is empty my model becomes flat and our can-delegate becomes the same as in RBDM0.

4.3.3   Revocation in RBDM1

We now turn our attention to the revocation part of RBDM1. Similar to revocation in RBDM0, my model has two approaches to implement revocation of previously delegated roles. In the first approach, it appends a lifetime to each delegation. Once that time expires, so does the delegation. The second approach my model uses to implement revocation is allowing users to revoke the memberships of delegated roles (human revocation).

The following sub-sections discuss these types of revocations and address some of the issues that might introduce complexity and subtlety to the model.

4.3.3.1 Revocation Using Time Outs

In this model, where the delegation is temporary and expires with time, the length of the delegation becomes critical to the effectiveness of delegation. This period, which I refer to in my model as duration of delegation, must be chosen carefully. Overestimating the duration of delegation increases risk by allowing the delegate member to continue to execute the permissions assigned to the delegated. Underestimating the duration of delegation might prevent the delegate member from completing the assigned task. The concept of delegation duration was explained in RBDM0.

4.3.3.2 Human Revocation

In the cases where revocations are implemented by humans, my model authorizes revocation under the following conditions:

- Any original member can revoke:

 This approach has some advantages and disadvantages. Among the disadvantages are:

- It does not allow the original delegating member to track and control the behavior of the temporary delegate member.

- It raises the possibility of conflicts between the original members that might result from having someone else other than the sponsoring original member revoking the delegated membership.

Among the advantages of this approach are:

- Protection of the system resources from the delegate member doe not depends solely on the delegating role member. If the delegate member behaves badly in the delegated role, then any original member in that role can revoke his membership, which does not take a long time.

This revocation approach raises some issues that introduce complexity and subtlety. The following discussion addresses these issues.

For the sake of illustration I have used Table 4.1, in conjunction with Figure 4.6, to discuss the revocation issues associated with the delegation in hierarchical roles.

Suppose that Alice, who is an original member of role PL1 (Alice $\in$ User_O(PL1)), delegates her membership to Bob who is an original member of role PE1 (Bob $\in$ User_O(PE1)), (PE1$\leq$ PL1). Thereby ((Bob, PL1) $\in$ UADE), and ((Bob, r') $\in$ UADI), where r' is any role that is junior to PL1 (PL1 $\geq$ r'). This is done at Alice's discretion

because Alice acts as an owner of role PL1 because of her original membership in that role. Alice can later revoke Bob's delegate membership of role PL1 (and from any role that is junior to PL1). Note that in this case, an original member of any role that is senior to role PL1 can also revoke Bob's membership in PL1. This is because original members of senior roles are also original (implicit) members of junior roles. In my example, this means Frank can remove Bob from PL1.

Now suppose that Bob was made a member of role PL1 by Alice, and by Dave, who is another member of PL1, not shown in Figure 4.6. If Alice revokes Bob's membership in PL1, then Bob should still continue to retain his membership in PL1, via Dave. Bob can be totally revoked from PL1 only if both Alice and Dave revoke his membership in PL1.

- Cascading Revocation

Cascading revocation refers to the way a delegation of membership can become automatically revoked as a result of the revocation of the membership of the roles involved.

My model supports cascading revocation. In the above example, suppose that Alice's membership of role PL1 is revoked by a security officer. This will result in the automatic revocation of Bob's membership in role PL1 (and from any roles junior to PL1). Also, if Bob loses his membership in his original role (PE1), this will lead to losing his delegate

membership of role PL1 (and any roles junior to role PL1). However, if Dave's membership in role PL1 was in turn given by Alice, then if Alice revokes Bob's membership of PL1, Bob will also lose his membership in role PL1 obtained from Dave. Alice can also revoke the membership of Bob in role PL1 indirectly by revoking Dave's membership of PL1.

- Multiple sponsoring / supporting roles

Multiple supporting roles are when a user who is an original member of more than one role gets delegated more than once to the same role, one for every role membership. This is also allowed in my model.

In both cases, the delegate member in a role is dependent of both the sponsor and the supporting roles. If either of these roles is revoked, the delegate membership will also end up being revoked.

4.4   Summary of TRBDM

In this chapter I have described the motivation, intuition, and outline of a new simple and non-trivial model for human-to-human delegation using roles called TRBDM (role-based delegation model with temporary delegation) that is based on the Role-Based Access control (RBAC96) developed by [SCFY96]. TRBDM has two main components:

RBDM0 (role-based delegation model using flat role), and RBDM1 (role-based delegation model using hierarchical roles). In the first section of this chapter, the first component (RBDM1) was described in full detail, and then in the second section I described the second component (RBDM1) to show how the flat-roles model can be extended to include hierarchical roles.

Chapter 5

ROLE-BASED DELEGATION MODEL WITH PERMANENT DELEGATION (PRBDM)

5.1   Introduction

In this chapter, I derived the second type of role-based delegation models to address the delegation cases produced using the framework for the role-based delegation model discussed in Chapter 3 of this dissertation. More specifically, in this chapter, I have derived two models to address the permanent, non-montonic, and total delegation between users who belong to different roles.

In real life there are some situations where a user who belongs to a certain role in an organization is no longer able to carry out his task. Thus, he wishes to permanently delegate his role to another user who belongs to a different role. For example: a professor who is a member of a doctoral advising committee goes on a sabbatical leave and is no longer able to serve as a member in the committee. This professor can delegate his role membership to another professor to serve as his permanent replacement in the advising committee role. As a result, the delegating professor loses all of his power in

the role, and the delegated professor will assume full power in the advising committee role.

In another situation, a user who is a member of a certain role and is assigned to a certain task might realize that another user who belongs to another role can better perform this task. In this case, the user may decide to permanently delegate his role to that other user. For example, a professor who is a member of a chairman role in a committee may realize that one of his committee members is better fit for that role. Therefore, provided that both the delegator and the receiver agree on the permanent delegation, the first user can permanently delegate his role to that committee member.

In chapter 3 of this dissertation, I demonstrated why permanent delegation is useful only if it is non-monotonic and total.

I have argued that permanent-monotonic delegation does not appear to be very desirable if a user in a role elected to delegate his or her role permanently to another user who is not a member of that role, making the delegated member a permanent replacement of the delegator. In such case there is no need for the delegating role member to maintain his or her power in that role. Doing so would increase the number of users in the delegated role.

I have also argued that permanent non-monotonic delegation, on the other hand, seems to be more useful. Once a delegating user permanently delegates his or her role to another user and loses his or her power in that role, then he or she is no longer responsible for the

behavior of the newly delegated member. Non-monotonic delegation allows the new role member to act independently and as if he or she is an original member of that role. For example, a member of an advising committee (say, Alice) decides that she can no longer serve in the committee and is willing to permanently delegate her role to someone else who is not a member of this committee (say, Bob). In this scenario, it makes sense that the delegation to Bob by Alice be non-monotonic. Alice loses all of her power in that role. In fact, to preserve the security of committee role, the delegation must be non-monotonic. This is because Alice relinquishes her membership of that role.

I further argued that when the delegation is permanent and non-monotonic, partial delegation is not desirable because delegating only a subset of the role does not allow the delegated member to carry out the full task of the delegating member. For example, suppose that Alice (who is an original member of Role a) permanently, and non-monotonically delegates only a subset of her role to Bob (who is a member of Role b). In this case, Bob will not be able to fully carry out the task of Alice (e.g., if the task of reviewing student records was not delegated, then Bob will not be able to provide efficient advice to the students). Moreover, neither Alice nor Bob will end up with full power in that role after the delegation. This is because upon delegation, Alice will not retain any power in the delegated role, and Bob will dose not acquire the full permissions of that role.

The important issue in modeling this type of delegation is that it does not violate RBAC's security principles: least privilege, separation of duties, and data abstraction.

With permanent delegations, only a security officer can revoke the membership of this new-delegated role.

In this chapter, I have derived two models to address permanent delegation between users who belong to different roles. In the first model, which is called Permanent Role-Based Delegation in Flat Roles (PRBDM0), I have addressed the delegation in the cases where the relationship between the involved roles is flat - no inheritance between the roles is considered. In the second model, which is called the Permanent Role-Based Delegation Model in Hierarchical Roles (PRBDM1), I have addressed delegation in the cases where the relationship between the involved roles is hierarchical.

Both models use the can-delegate relation, which controls the way in which delegation between users in different roles is authorized. In permanent delegations, only the security officer can revoke the membership of users from their roles, so the semantics of revocation is straightforward.

The next section discusses the role-based delegation model that is based on permanent delegation where the relationship between the involved roles is flat.

5.2 Permanent Role-Based Delegation Models in Flat Roles (PRBDM0)

This model is another form of the RBDM models and is based on RBAC96. It is formalized to implement the delegation path of permanent, non-monotonic, self-acted, and total delegation. This means that the delegation addressed here can be characterized as follows: 1) the delegation is between users in flat roles (no inheritance of permissions between roles is involved); 2) upon delegation, the delegating member loses his power in that role; 3) the delegation is administered by the delegation member himself; and 4) the delegation is total (no partial delegation is allowed). Next we give some assumptions and basic element of PRBDM0:

5.2.1 Assumptions and Basic Elements

Delegation between members in the same role is not allowed because it is meaningless.

The delegation is total. Each user in a delegating role delegates the total package of permissions embodied in that role or does not delegate at all.

Upon delegation, the delegate members permanently assume all permissions in the delegated role, and hence can be considered permanent members. In this model, revocation is not relevant simply because the delegation is permanent and non-

monotonic. This means that the delegating member can no longer revoke the roles he delegated. Therefore, only the security officer handles revocation of memberships.

The time element associated with delegation, which I referred to in other models as "duration," is not relevant in this model. Delegation in this model is permanent. Thus, there is no time limit for its life. The security officer handles membership revocation.

More specific assumptions include the following:

- Only the original role members can delegate.

- Upon delegation the delegator will no longer be accountable.

- The delegate member will not be under supervision.

- Delegation is permanently assigned.

- Delegate members will have identical rights as the other original members in the role; in other words, they become original members.

- The security officer can reinstate the original member to his role.

The following definitions formalize the above discussion:

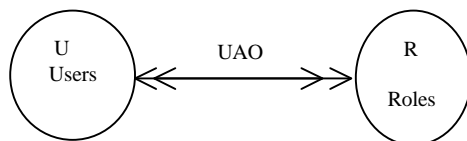**Definition 5.1:** The PRBDM0 model has the following components:

Figure 5.1. PRBDM0

1. $UAO \subseteq U \times R$ is a many-to-many, original member to role assignment relation

2. UA = UAO All members of a role become original members

5.2.2 Delegation in PRBDM0

In RBAC96, the security officer handles assignment of users to roles [San96]. In PRBDM0, the delegation from one user in a delegating role to another user in a different role is actually making the delegated user a member of the delegated role (this membership is also original). Thus, the delegating user handles this function. This function is a widely decentralized task that can be taken care of by the users themselves without continuous involvement from the security officer.

It is important to note that in PRBDM0, the notion of delegate membership no longer applies. All members, whether they were originally assigned to a role by the a security officer, or were brought in by other original member that are no longer want to be members of that role, are considered to be original.

Role-role delegation is authorized in PRBDM0 using the following relation:

Definition 5.2: PRBDM0 controls role-role delegation by means of the relation can-

delegate $\subseteq R \times R$.

Can-delegate is irreflexive. This means that a user in a role cannot delegate his

membership to another user in the same role since this is meaningless.

The meaning of $(a, b) \in$ can-delegate is that a user (say, Alice) who is an original

member of role a can delegate her role membership to any another user (say, Bob) who is

an original member of another role b. This results in Bob becoming an original member

of role a and Alice losing her membership in role a. For example, If Alice $\in$ User_O(a)

and Bob $\in$ User_O(b), then Alice can delegate a to Bob, and so thereby Bob $\in$

User_O(a), and Alice $\notin$ User_O(a).

5.2.3   Revocation in PRBDM0

In permanent, non-monotonic delegation, revocation by the delegator is not relevant

because, under this scenario, the delegator loses all of his power over the delegated role.

Also, upon delegation the delegate role member will get the same permissions as any

permanent member in that role. Therefore, only the security officer can revoke the

membership of the new member. Moreover, only the security officer can reassign the

original delegating role member to his role, or some other member can make him a new permanent member in that role via delegation.

5.3 Permanent Role-Based Delegation Models in Hierarchical Roles (PRBDM1)

The next model of delegation in this chapter is Permanent Role-Based Delegation Model in Hierarchical Roles (PRBDM1). This model is an extension to the PRBDM0 which was discussed in the previous section of this chapter. The main addition in this model is that it introduces role hierarchies to the previous model.

In this section, I formally define a role-based delegation model based on hierarchical relationship between the roles involved. I also identify the different semantics that impact the can-delegate relation, analyze these semantics to determine which ones I consider as more appropriate in business today (thus allowed in my model) and provide a justification as to why those selections are made.

First, I give some assumptions and basic elements:

5.3.1 Assumptions and Basic Elements

This model is an extension to PRBDM0. Thus, in addition to the basic assumptions stated in the previous model, this model adds the following constraints:

- Delegation can only be either downward or cross, for the reasons given below.

- Upward delegation is useless because senior roles inherit all the permission of their junior roles.

- Downward delegation means that a user who is an original member of a delegating role delegates to other users who are original members of roles that are junior to the delegating role. This type of delegation is appropriate for promoting members who belong to qualified junior roles to be members in senior roles. In the case of permanent delegation, the delegation has to be total.   This is because, upon delegation, the delegator will lose all of his power inside the delegated role, and if the delegate member does not receive the entire role, then he/she will not be able to execute all tasks assigned to that role. Moreover, with permanent partial delegation, no one gets full membership in the delegated role, which makes this type of delegation not very useful.

- Cross delegation means that the delegation takes place between users who are members of incomparable roles. For example, a manager in the sales department who is also a member of the auditing department may feel that another user who belongs to another department (say the security department) is better fit for the auditor role than him. In this case, the sales department manager may elect to permanently delegate his membership in the auditing department to the other user who is a member of another role (the security department). In this case, the delegate member will assume the full power in the auditing department and the sales department

manager (delegator) will lose all of his power in the delegated role (the auditing department).

The following definitions formalize the above discussion.

Definition 5.3 The following is a formal definition of PRBDM1:

The definition of PRBDM1 is the same as PRBDM0, with the following elements being added (see figure 5.3):

- $RH \subseteq R \times R$ is a partial order (this can be written as $\geq$ in infix notation). Also, the less familiar symbol $||$ is used to denote non-comparability: we write $x||y$ if $x \neg \leq y$ and $y \neg \leq x$.
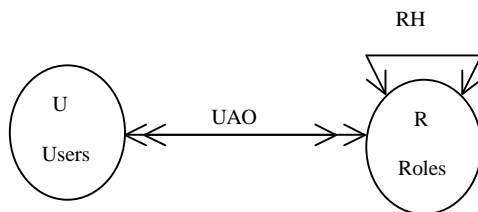

Figure 5.2. PRBDM1

5.3.2 Delegation in PRBDM1

In PRBDM1, authorization of delegation depends on the semantics of the can-delegate relation. These semantics become especially complicated when the membership statuses of the delegating and the delegated roles vary from one situation to another. For example,

a delegation by an original explicit delegating role to an original implicit delegated role will carry a different meaning than a delegation by an original implicit role that delegates to an original explicit role, and so on.

In RBDM1, I took a very liberal approach by allowing the can-delegate relation to use all different combinations of original explicit and implicit memberships. In PRBDM1, I have taken a more conservative approach when choosing which combinations are authorized by the can-delegate relation. This is because of the permanent nature of the delegation. As we will see later in this chapter, once a delegating user completes the non-montonic delegation to another user, the delegator relinquishes his power in the delegated role and the new member becomes an original member of the newly delegate role. This did not have a major impact in PRBDM0, but in PRBDM1, this will have a major impact on the model.

Definition 5.5 The user-role assignment is authorized in PRBDM1 by the following relation: can-delegate $\subseteq R \times R$

Can-delegate is irreflexive. This means that a user in a role cannot delegate his membership to another user in the same role since this is meaningless. This is restriction especially important in PRBDM1, because allowing a reflexive delegation will reduce the number of members in the role. Also, $(a, b) \in$ can-delegate implies $a > b$ or $a \parallel b$.

The meaning of (a, b) ∈ can-delegate varies depending upon the membership status of the delegating and the delegated roles at the time of delegation. The following section describes this concept.

5.3.2.1 Semantics of Delegation in PRBDM1

In this section, I list and analyze the different semantics that impact delegation in PRBDM1 and explain the approach my model takes toward selecting the appropriate semantics of delegation.

I use the same example from RBDM1 to discuss the different semantics of delegation in PRBDM1. Also, for the sake of my illustrations, I refer to Figure 4.6, and Table 4.1.

I use OE to denote original explicit members and OI to denote original implicit members. Hence the four possibilities are (OE, OE), (OE, OI), (OI, OE), and (OI, OI), where the first item of each tuple represents the delegating role member and the second represents the delegated role member.

In RBDM1, where the delegation is temporary, my approach was most liberal; I allowed all different combinations to be authorized. Due to the non-monotonic nature of this delegation in PRBDM1, here, I have chosen a more conservative approach. In this model, where the delegation is permanent, and upon delegation, the delegator loses his control in the delegated role, a more tightly controlled authorization is required. Therefore, in this

model, I have allowed only two combinations (OE, OE) and (OI, OE). The following discussion explains the reason behind this approach:

Using Figure 4.6, Table 4.1, and the examples of RBDM1, I provide the following arguments:

In the first scenario where the delegating role is original explicit and the delegated role is also original explicit (OE, OE), (PL1, E1) ∈ can-delegate means the following:

1. Alice can delegate PL1 to Dan, Alice can delegate PE1 to Dan, and Alice can delegate QE1 to Dan. In this case, Alice relinquishes her power in PL1, PE1, and QE1 respectively.

   In this scenario, it is not desirable for Alice to permanently delegate PE1 or QE1 to Dan without delegating PL1 to Dan. This is because by delegating PE1 or QE1 to Dan, Alice will relinquish her power in the respective roles only, and maintaining her membership in PL1, thus violating the inheritance rule of the role hierarchy.

2. Alice cannot delegate PL1 to Bob or to Charlie, because Bob and Charles are not explicit members of E1.

3. Finally, Alice cannot delegate PL1 to Frank, because this is meaningless

In the second scenario, where the delegating role is original explicit and the delegated role is original implicit (OE, OI), (PL1, E1) ∈ can-delegate means the following:

Alice can delegate PL1 to Dan, Alice can delegate PL1 to Bob, and Alice can delegate PL1 to Charlie. This is allowed under the delegation rule. However if we look at it from preserving the separation of duty principle, which is one RBAC96 main features, we see that by allowing Alice to permanently delegate PL1 to Bob, Bob become a permanent member of both PL1 and PE1. This might lead to a conflict of interest, since Bob's membership in PL1 gives him an advantage over Charlie, who is a member of a non-comparable role. To satisfy both rules (delegation rule and separation of duty role) this can be handled by adding some restriction on this delegation, i.e. Bob can be a permanent member of PL1 only if he relinquishes his role in PE1. Similarly, this applies to the case where Alice delegates PL1 to Charles

In the third scenario, where the delegating role is original implicit and the delegated role is original explicit (OI, OE), (PL1, E1) ∈ can-delegate means the following:

Frank can delegate PL1 to Dan, Frank can delegate PE1 to Dan, and Frank can delegate QE1 to Dan. This is not desirable, because all three cases will lead to Frank losing his membership in PL1, PE1, and QE1, which violate the inheritance rule in the hierarchy. Therefore, this combination will not be allowed in this model

Finally, in the fourth scenario, where the delegating role is original implicit and the delegated role is also original implicit (OI, OI), (PL1, E1) ∈ can-delegate means the following:

1. Frank can delegate PL1 to Dan, Frank can delegate PE1 to Dan, and Frank can delegate QE1 to Dan. This is not desirable, because all three cases will lead to Frank losing his membership in PL1, PE1, and QE1, which violate the inheritance rule in the hierarchy.

2. Frank can delegate PL1 to Bob and to Charlie. This also leads to Frank loosing his membership in PL1, which also violates the inheritance rule of the role hierarchy.

Therefore, this combination will not be allowed in this model

In conclusion, in order to making this model operate gracefully, I restrict the semantic of delegation to only the explicit and very limited implicit cases of role memberships. More specifically, in PRBDM1, only (OE, OE), and (OE, OI) semantics are allowed in the can-delegate relation of this model.

Therefore, after restricting the delegation to only the (OE, OE) and (OE, OI), Table 4.1 of RBDM0 will be reduced to the following table (Table 5.1) in PRBDM1.

Table 5.1. Examples of Authorization Functions in PRBDM1

	Status of the role memberships		Given that (PL1, E1) Î can-delegate
	Delegating role	**Delegated role**	Semantics of can-delegate relations
PRBDM1 (Hierarchic al roles)	OE	OE	Alice cannot-delegate PL1 to Bob Alice cannot-delegate PL1 to Charlie Alice can delegate PL1 to Dan
	OE	OI	Alice can-delegate PL1 to Bob Alice can-delegate PL1 to Charlie Alice can delegate PL1 to Dan

5.3.3   Revocation in PRBDM1

In PRBDM1, only the security officer can revoke the permanently delegated role memberships. Once a user who is a member of a role becomes a permanent member of another role, that user assumes all the privileges/permissions that the original members of the delegated role have. Also, upon delegation, the delegating user loses all of his power in the delegated role. Hence, no one has the authority to revoke the new member's membership in the delegated role.

5.3.4   Summary of PRBDM

In Chapter 4 I have described the motivation, intuition, and outline of a new simple and non-trivial model for human-to-human delegation using roles called TRBDM (role-based delegation model with temporary delegation). In this chapter, my focus turned to addressing permanent delegation. This was accomplished by developing a role-based

delegation model that was called PRBDM. PRBDM has two components. The first component, which is called Permanent Role-Based Delegation in Flat Roles (PRBDM0), addresses the delegation in the cases where the relationship between the involved roles is flat - no inheritance between the roles is considered. The second model, which is called the Permanent Role-Based Delegation Model in Hierarchical Roles (PRBDM1), addresses delegation in the cases where the relationship between the involved roles is hierarchical. In PRBDM1, authorization of delegation depends on the semantics of the can-delegate relation. In this model I give a can-delegate relation, but I restricted the use of the semantics of delegation to only the ones that do not violate the tight control principle of PRBDM1.

Chapter 6

CONCLUSION

This chapter enumerates the contributions of this dissertation and discusses future directions. Section 6.1 presents the different contributions and section 6.2 gives future research directions.

6.1   Contributions

In the information security arena, one of the most interesting and promising techniques proposed is Role-Based Access Control. In the last few years, much work has been done in the definition and implementation of RBAC. However, this recent work has not yet included the concept of delegation in RBAC. Delegation in computer systems can be human-to-human, human-to-machine, machine-to-machine, and perhaps even machine-to-human. Most of these types of delegations have received some attention in the literature; however, the concept of human-to-human delegation has not yet been addressed.

In this dissertation I have focused on human-to-human delegation in computer systems. Specifically, I have developed a series of simple but practically useful models for delegation, in which a user can use RBAC philosophy to delegate his or her role to

another user who belongs to another role. This research is the first attempt to address this type of delegation.

Performing human-to-human delegation within the framework of RBAC contributes to the evolution of RBAC, adding to the already good reputation of RBAC, and gives us a simple and effective way to address the concept of delegation between humans.

This dissertation has shown that it is possible, by adding a can-delegate relation to the RBAC in conjunction with constraints, to produce a framework for role-based delegation models. The research approach used for this purpose was an exploratory approach.

The contributions of this dissertation can be summarized in the following:

1) My first contribution in this dissertation is that I have provided a framework for role-based delegation models. This was accomplished by identifying the characteristics related to delegation, using these characteristics to generate possible delegation cases, and using a systematic approach to reduce the large number of cases into a few cases, which can be used to build role-based delegation models.

2) My second contribution is that I developed two role-based delegation models to illustrate some practical access control policies. These policies were illustrated

both in flat and hierarchical roles. I showed that using a relation, called can-delegate, a user who belongs to a certain role can delegate his/her role to another user who belongs to another role. I also showed that the delegated role can be revoked.

3) I have analyzed the different properties of the can-delegate relation, with each model individually as well as with the two models combined. In the case of the flat relation between the roles, can-delegate is straightforward, involving only two roles. However, in the hierarchical relations, can-delegate is more complicated -- it involves the two roles as well as their junior roles. Agent-based delegation does not impact the can-delegate relation, thus was discussed briefly in the future work section.

6.2   Future Work

Based on the research work done in this dissertation, I propose the following future research directions and issues.

6.2.1   Developing an Agent-Based Role Delegation Model

In Chapter 3 of this dissertation, I developed a framework for role-based delegation models and identified some interesting cases that showed that human-to human

delegation can occur in many different ways: Permanent, Temporary, One step, Self-acted, etc. I have also developed some models for delegation, in flat roles and in hierarchical roles, where the delegating user himself always administers the delegation.

Future work should focus on agent-based delegation. Specifically, it should address how a third party, called an agent, can administer the delegation on behalf of a user who is a member of a certain role and wishes to delegate his role to another user who belongs to another role.

Agent-based delegation is motivated by the fact that in real life, there are cases where a user who is a member of a certain role in an organization needs to delegate his/her role to some other user who is a member of a different role to complete some task. This can be accomplished using a third party (an agent) whose responsibility is only to administer the delegation between users in different roles, in the cases when the actual delegating role member is not available. For example, in physical space, a secretary of a department in a university can be given the keys to all professors' offices, so that if any of the professors is not available to complete a certain task, and someone else (such as a teacher assistant or lab assistant) needs access to one of the professor's rooms in order to complete the task on behalf of the professor, the secretary can administer the delegation for the professor and give the key to the delegate to access the professor's room. In cyberspace, where the access to the system resources is controlled by the role memberships, an agent role can be

defined to administer the delegation of that same professor's role to someone else in order to complete a task on behalf of the professor.

There are self-delegations addressed in the literature, where a user can go out and grab a membership in a role to accomplish a certain task and then, once that task is complete, relinquish that membership. I consider this type of delegation as very dangerous. My view treats delegation a little tighter than that of the self-delegation, by adding some restrictions.

I have identified two different manifestations in which an agent-based role delegation can be done: Role-participant agent and Non-role participant agent. These two manifestations can be further extended to include dynamic and static types of delegations. The table below shows these manifestations.

Table 6.1. Framework for Agent-Based Role Delegation

	Role Participant Agent	**Non-Role Participant Agent**
Dynamic Delegation	ABRADM-DRP	ABRDM-DNRP
Static Delegation	ABRADM-SRPA	ABRADM-SNRPA

In summary, future work should include describing a framework of reference to systematically address the diverse manifestations of the agent-based delegation model,

and developing an agent-based delegation model that illustrates delegation based on non-role participant delegation. Other models to illustrate the other manifestations should be similarly developed.

6.2.2   Implementation of the RBDM Models

I would like to see a tool built to validate these delegation models. This can be done in two phases: the first phase is to use the model in its current state and then the second phase to incorporate other models of RBAC 96 (RBAC2 and RBAC3).

6.2.3   Extension of RBDM

I would like to see the role-based delegation concept that was addressed in this dissertation be extended further to include the other components of the RBAC96. I believe this type of extension would make the research on the human-to-human using roles more complete.

1.   Adding Multi-Level Delegation

In this dissertation, I have limited my delegation to a single step, and, in some cases, a two step delegation. The reason is that most of the literature suggests that multi-step delegation/assignment is a very complicated issue. Therefore, I elected to limit my work to the single step in order to keep my models simple.

I should mention that other authors have taken my work and used it as a foundation for addressing role-based delegation in multi-step delegation [ZAC01].

2.   RBDM Within the Framework of RABC2

I would like to apply constraints to role-based delegation models. This means that the work will expand to include the RBAC2 (the constraints model of RBAC 96).

In some parts of this dissertation I did apply minimum constraints in order to preserve the security of RABC. However, the full constraint concept of RABC2 was not addressed in this dissertation. I believe researching role-based delegation with the framework of RABC2 can provide a stronger control on handling delegation between humans. The reason for not addressing this concept in my dissertation is, first, I considered delegation between humans to be a discretionary act, and hence, I wanted to keep it simple. The other reason is that RBAC2 adds a whole different dimension to role-based delegation, which may require considerable time and effort. Therefore, I believe this concept to be a topic for another research effort.

I believe that even though delegation is discretionary, in some situations, where more control on the role assignment and delegation of these roles are needed, constraints can be a powerful tool for enforcing that control.

3. RBDM within the Framework of RBAC3 (the Consolidated Model)

I would like to see the concept of delegation between humans be researched within the framework of the consolidated model of RBAC 96.

6.2.4   Integrating RBDM with PKI

I would like to see the concept of role-based delegation discussed in this dissertation be integrated with other technologies such as the Public Key Infrastructure (PKI) where a digital certificate can be delegated between users on the system. I believe that a marriage can take place between the role delegation concept discussed in this dissertation and PKI. I believe that PKI can use the same concept of role-role delegation discussed in this dissertation to delegate certificates and revoke certificates between users that belong to different domains, provided that these users are part of the same certification authority.

References

References

[ABLP96] Martin Abadi, Michael Burrows, Butler Lampson and Gordon Plotkin. A calculus for Access Control in Distributed Systems. ACM Transactions on Programming Languages and Systems, Vol. 15, No 4, September 1993, pages 706-734.

[BS2000] Ezedin Barka and Ravi Sandhu. Framework for Role-Based Delegation Models. In Proceedings of 16th Annual Computer Security Application Conference, New Orleans, LA, December 11-15 2000, pages 168-176

[BS2000] Ezedin Barka and Ravi Sandhu. A Role-based Delegation Model and Some Extensions. Proceedings of 23rd National Information System Security Conference, pages 101-114, Baltimore, Oct.16-19, 2000

[FCK95] David Ferriaolo, Janet Cugini, and Richard Kuhn. Role-based access control (RBAC): Features and motivations. In Proceedings of 11th Annual Computer Security Application Conference, pages 241-48, New Orleans, LA, December 11-15 1995.

[FK92] David Ferriaolo and Richard Kuhn. Role-based access controls. In Proceedings of 15th NIST-NCSC National Computer Security Conference, pages 554-563, Baltimore, MD, October 13-16 1992.

[Glad197] Henry M. Gladny, Access Control for Large Collections. ACM Transactions on Information Systems, Vol.15, No.2, April 1997, Pages 154-194.

[GM90] Morrie Gasser, Ellen McDermott. An Architecture for practical Delegation in a Distributed System. 1990 IEEE Computer Society Symposium on Research in Security and Privacy. Oakland, CA. May 7-9, 1990.

[HRU76] M.H. Harrison, W.L. Ruzzo, and J.D. Ullman, Protection in Operating Systems. Communications of ACM. 1976. Pages 461-471.

[HSP01]   Asa Hagstorm, Sushil Jajodia, Francesco Parisi-presicce, Revocations- a Classification. 2001 IEEE Computer Society Symposium on Research in Security and Privacy. Oakland, CA. May 7-9, 2001.

[Lamp71] B.W. Lampson, Protection. 5th Princeton Symposium on information science and systems. Pages 437-443.

[San92]   Ravi Sandhu, The Typed Access Matrix Model. Proceeding Symposium on Security and Privacy, Oakland, CA, May 4-6, 1992, pages 122-136.

[San97]   Ravi Sandhu. Rationale for the RBAC96 family of access control models. In Proceedings of the 1st ACM Workshop on Role-Based Access Control. ACM, 1997.

[SBM99] Ravi Sandhu, Venkata Bhamidipati and Qamar Munawer. "The ARBAC97 Model for Role-Based Administration of Roles." *ACM Transactions on Information and System Security*, Volume 2, Number 1, February 1999, pages 105-135.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. IEEE Computer, 29(2):38-47, February 1996.

[VAS91]   Vijay Varadharajan, Philip Allen, Stewart Black. An Analysis of the Proxy Problem in Distributed systems. IEEE Symposium on Research in Security and Privacy. Oakland, CA 1991.

[ZAC01]   Zhang, Ahn, and Chu, A Rule-Based Framework for Based Delegation. Proceeding of the 6th ACM Symposium on Access Control Models and Technologies, Pages 153-162, Chantilly, VA, May 3-4, 2001.

CURRICULUM VITAE

Ezedin E. Barka was born on September 09, 1959, in Tripoli, Libya, and is an American citizen. He graduated from Tripoli High School, Tripoli, Libya, in 1977. He received his Bachelor of Science from Indiana University in 1983, and his Master of Arts in Public Administration/Management Information Systems from the University of Maryland-University College in 1991. He is currently employed as a Senior Network Security Specialist by Signal Corporation.