uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Wei Qiu

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Exploring User-to-Role Delegation in Role-Based Access Control

TITRE DE LA THÈSE / TITLE OF THESIS

Carlisle Adams

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Thomas Tran

Anil Somayaji

Gary W. Slater

Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Exploring User-to-Role Delegation in Role-Based Access Control

Wei Qiu

Thesis Submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master of Computer Science

Under the auspices of the Ottawa-Carleton Institute for Computer Science

uOttawa

L'Université canadienne
Canada's university

University of Ottawa
Ottawa, Ontario, Canada

November 2006

Your file Votre référence
ISBN: 978-0-494-25824-8
Our file Notre référence
ISBN: 978-0-494-25824-8

Abstract

Mechanisms must be provided to protect resources from attackers when users request access to resources in network environments. Role-Based Access Control (RBAC) formulates that access decisions are based on the roles that individual users have as members of a system. In RBAC, there are role hierarchies in which a senior role inherits the permissions of a junior role. In order to allow a junior role to perform one or more tasks of a senior role, various delegation models have been proposed in the literature, including Role-Based Access Control Model (RBAC96), Role-Based Delegation Model (RBDM0), Attribute-Based Delegation Model (ABDM), Role-Based Delegation Model 2000 (RDM2000) and Permission-Based Delegation Model (PBDM).

The main work of this thesis presents a flexible conceptual delegation model called User-to-Role Delegation Model (URDM), which is based on RDM2000. URDM supports role hierarchy, single-step delegation and simultaneous delegation by introducing a new delegation relation. Four situations are addressed when URDM is involved. We also implement a web application named University Delegation Management System (UDMS) for URDM.

At the end of the thesis, we make some generalizations to the area of role-based delegation in access control and present directions for future research.

Acknowledgments

To my dear dad, Xingruo Qiu and mom, Qiying Zhu.

My heartfelt gratitude to my supervisor, Dr. Carlisle Adams, for his insight, guidance, and encouragement throughout this thesis work. He is not only one of the best professors in my life, but also one of the greatest experts in access control research area I fortunately met and learnt a lot from.

I also send my sincere appreciation to my dear sister, Rong Qiu and her husband, Liping Sun for their love and support.

Many thanks to Information Security Research Group (ISRG). Our monthly meetings were of great help.

Thanks to Ding Cai, for providing a software tool named Dreamweaver 8.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ABDM	Attribute-Based Delegation Model
ACL	Access Control List
ADO	ActiveX Data Objects
AP	Administrative Permissions
AR	Administrative Roles
ASP	Activate Server Pages
CBAC	Coalition-Based Access Control
CR	Prerequisite Condition
CU	Current Users
DAC	Discretionary Access Control
DAE	Delegation Attribute Expression
DD	Delegation Depth
DIR	Director
DML	Data Manipulation Language
DR	Delegated Role
DSI	Dynamic Systems Initiative
DT	Delegation Time
DTR	Delegation Roles
DW	Delegation Width
EE	Electronic Engineering
FDBR	Fixed Delegatable Roles
FSM	Finite State Machine
GD	Grant-Dependent Revocation
HTML	Hyper Text Markup Language
I&A	Identification and Authentication
IIS 5.1	Internet Information Services 5.1
LAN	Local Area Network

MAC Mandatory Access Control

OU Original Users

P Permissions

PA Permission Assignment

PAD Permission-Delegation Role Assignment

PAR Permission-Regular Role Assignment

PBDM Permission-Based Delegation Model

PC Delegation Prerequisite Condition

PDF Post-Doctoral Fellow

R Roles

RA Research Assistant

RAE A Set of Attribute Expressions for Revocation Requirements

RBAC Role-Based Access Control

RBAC96 Role-Based Access Control 96 Model

RBDM0 Role-Based Delegation Model 0

RDM2000 Role-Based Delegation Model 2000

RH Role Hierarchy

RR Regular Roles

RUAC Rule-Based Access Control

S Session

SE Software Engineering

SD Single Delegation

SDR Simultaneous Delegation Relation

SP Subset Permissions

SQL Structured Query Language

STD Single Step Delegation

T A Set of Durations

TBAC Task-Based Access Control

TDBR Temporal Delegatable Roles

TMAC Team-Based Access Control

U	Users
UA	User Assignment
UAD	Delegate Member to Role Assignment
UAO	Original Member to Role Assignment
UDMS	University Delegation Management System
URDM	User-to-Role Delegation Model
WAN	Wide Area Network

Chapter 1 Introduction

1.1 Introduction

Access control is the process by which users are identified and granted certain privileges to information, systems, or resources. Understanding the basics of access control is fundamental to understanding how to manage proper disclosure of information in a computer networking environment.

The domain of this research is in the field of role engineering for Role-Based Access Control (RBAC). RBAC has been considered a promising technology for managing and enforcing security in large-scale distributed systems.

The subsequent sections of this chapter will cover the following topics:

- Overview of Thesis Work
- Thesis Related Work
- Thesis Contributions
- Thesis Outline

1.2 Overview of Thesis Work

Computer network systems, wired and wireless, Local Area Network (LAN) and Wide Area Network (WAN), public and private, are used daily to conduct transactions and communications among businesses, government agencies and individuals. The networks are comprised of nodes, which are individual user PCs and one or more servers and/or host computers. They are linked by communication systems, some of which might be private, such as within a company , and others that might be open to public access. The obvious example of a network system that is open to public access is the Internet, but many private networks also utilize publicly-accessible communications. Today, most companies' host computers can be accessed by their employees whether in their offices over a private communications network, or from their homes or hotel rooms through various ways. These employees may require different levels of access to different areas of LAN at different times for different business purposes. However, many corporations leave their virtual doors open and their virtual windows unlocked, providing unrestricted access to a variety of end users. That lack of infrastructure presents little challenge to any malicious users and is one reason that 80 percent of corporations surveyed in the 2003 FBI/Computer Science Institute Computer Crime and Security Survey reported internal security incidents. This means network security has become a serious issue. Network security involves all activities that organizations, enterprises, and institutions undertake to protect the value and ongoing usability of assets and the integrity and continuity of operations. An effective network security strategy requires identifying threats and then choosing the most effective set of tools to combat them. Enterprises must have business security solutions that provide detection and enforcement at every point of network access. To that end, corporations need a comprehensive, strategic approach to **Access Control**.

"Access control is the ability to permit or deny the use of an object (a passive entity, such as a system or file) by a subject (an active entity, such as an individual or process). Access control systems provide the essential services of Identification and Authentication (I&A), authorization and accountability, where identification and authentication

determine who can log onto a system, authorization determines what an authenticated user can do, and accountability identifies what a user did." [Wikipedia 2006b]

Traditional access control technologies can be categorized as either Discretionary Access Control (DAC) or Mandatory Access Control (MAC). DAC is a mode in which the creators or owners of files assign access rights, and a subject with discretionary access to information can pass that information on to another subject. MAC is an access policy determined by the system, not the owner. MAC is required for the mediation of all accesses of objects on the system. Since the access control system mediates all access to objects using rules imposed externally, users cannot give away permissions for object access. [Ferraiolo et al. 2003]

Role-Based Access Control (RBAC) is an approach for restricting system access to authorized users. It is considered an alternative approach to MAC and DAC.

RBAC determines access rights that roles can perform, and assigns users to roles. Users can access objects according to the related roles. As a result, the organization not only can preserve the access control policy appropriate to its characteristics consistently, but also can maintain an access control relationship between subjects and objects independently. The central functionality of RBAC is that access control policies for enterprise objects are written in terms of roles (i.e., collections of privileges), and each user in the environment is assigned to one or multiple appropriate roles. The major components of RBAC are roles, permissions, constraints and role-hierarchies [Coyne 1996]. A role is a logical grouping of atomic activities according to the physical constraints of the operational environment of the target system [Role Definition 2006]. This role normally contains all rights needed for a specific job function. Users can be easily assigned to these roles or reassigned from one role to another, acquiring the roles' permissions. Constraints refer to requirements for enforcing Separation of Duties (SOD), least privilege, conflict of interest and classification. A hierarchy is mathematically a partial order defining a seniority relation between roles, whereby senior roles obtain the permissions of junior roles.

RBAC greatly simplifies management of authorization while providing great flexibility in specifying and enforcing enterprise-specific protection policies and reducing the management costs. Users can be assigned to roles as determined by their responsibilities and qualifications. They can be easily reassigned without modifying the underlying access structure.

However, in RBAC, a junior role, or a role which is not included in the hierarchy, cannot perform the tasks of a senior role. In order to allow this when required, a delegation process is required to allow junior roles to be temporarily granted senior roles' permissions. Delegation of authority is in context of RBAC. The basic idea of delegation is that a delegating entity in a specific system can assign its permission(s) to another delegated entity. Existing delegation types in a computing environment can be human-to-human, human-machine, and machine-to-machine. Some delegation models in the recent literature address user-to-user delegation models, such as Role-Based Delegation Model 0 (RBDM0), Role-Based Delegation Model 2000 (RDM2000), Permission-Based Delegation Model (PBDM) and Attribute-Based Delegation Model (ABDM) [Barka et al. 2000], [Zhang et al. 2001], [Zhang et al. 2003], and [Ye et al. 2004].

RDM2000 has been proposed to support role hierarchy and sub-delegation. However, one challenge remains to be explored in RDM2000: In many cases, a delegating role member needs to delegate his or her permissions to more than one person simultaneously (to a role). For example, a professor may need to delegate his or her access right to a certain database to more than one of his or her research assistants at the same time. This type of delegation is referred to as the **simultaneous delegation problem**. The simultaneous delegation problem is a big issue in a large-scale environment: existing delegation models can only handle a single delegation process one by one; if there are one million delegation processes that have to be operated simultaneously, these existing models are not efficient.

In this thesis, a role-based delegation model called User-to-Role Delegation Model (URDM) is proposed. URDM is an extension of RDM2000, which supports simultaneous

delegation, single-step delegation, and role hierarchy. URDM is designed to handle simultaneous delegation in four situations:

- *Situation 1*: The delegating user can not delegate arbitrary subsets of permissions but can only delegate the entire role at a time to a set of delegated users, who are members of the same role before delegation.

- *Situation 2*: The delegating user can delegate subsets of permissions associated with the delegated role at a time to a set of delegated users, who are members of the same role before delegation.

- *Situation 3*: The delegating user can only delegate an entire role at a time to a set of delegated users, who are members of different roles before delegation.

- *Situation 4*: The delegating user can delegate subsets of permissions associated with the delegated role at a time to a set of delegated users, who are members of different roles before delegation.

Revocation is a reverse process of delegation. URDM also supports Grant-Dependent Revocation.

Because no formal processes have been addressed in the delegation models of RDM2000, PBDM and ABDM, we introduce delegation and revocation processes in URDM using Finite State Machine.

In order to demonstrate URDM, we design and implement a delegation system called University Delegation Management System (UDMS) by using a web design tool called Dreamweaver 8. We show how to handle *situation 1* by URDM in UDMS and provide a sample of implementation results.

1.3 Related Work

Delegation is an important component of access control. Researchers have proposed various delegation models. The basic idea of a delegation model is illustrated in Figure 1-1 [Yialelis and Sloman 1996]. When A invokes an operation op_1 on a target B, this operation triggers the invocation of op_2 on C. If A has the right to invoke op_2 on C but B has no right to invoke op_2 on C, then A temporally delegates the necessary access rights to B to invoke op_2 on C. In this case, A acts as a delegating user, B is considered as the delegated user, and C as the endpoint.

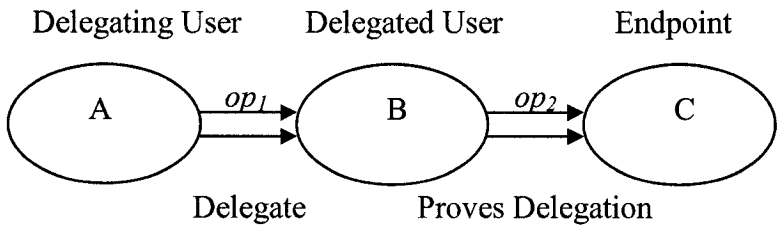

Figure 1-1. A Basic Delegation Model

In recent years, various models have been developed for different purposes, such as RBAC96, RBDM0, RDM2000, PBDM and ABDM. The development relationship between URDM and these five models is shown in Figure 1-2.

RBAC96 [Sandhu 1996a] is a delegation model family based on five sets of entities called users (U), roles (R), and permissions (P), and their administrative counterparts called administrative roles (AR) and administrative permissions (AP). Administrative roles and administrative permissions be respectively disjoint from the regular (i.e., non-administrative) roles and permissions in RBAC96. Moreover regular permissions can only be assigned to regular roles and administrative permissions can only be assigned to administrative roles. The RBAC96 family consists of RBAC with respect to regular roles and permissions ($RBAC_0$, $RBAC_1$, $RBAC_2$ and $RBAC_3$) and RBAC with respect to administrative roles and permissions ($ARBAC_0$, $ARBAC_1$, $ARBAC_2$, $ARBAC_3$). $RBAC_0$ is a base model for RBAC. Both of $RBAC_1$ and $RBAC_2$ include $RBAC_0$. $RBAC_1$ adds the

concept of role hierarchies, $RBAC_2$ adds constraints. $RBAC_3$ includes $RBAC_1$, and $RBAC_2$.

RBDM0 is the simplest form of the RBDM model and is based on $RBAC_0$ of the RBAC96 family. The delegation is between users in flat roles; no inheritance of permissions between roles is involved.

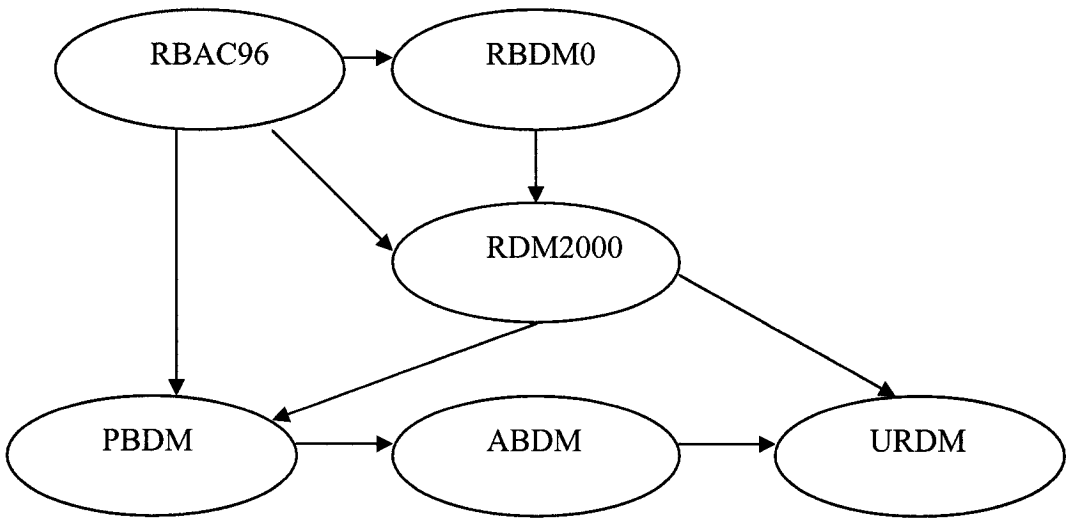

Figure 1-2. The Development Relationship between Models

PBDM supports user-to-user and role-to-role delegations with features of sub-delegation and various revocation mechanisms. In PBDM, a security administrator specifies the permissions that a delegating user has authority to delegate to others (delegated user), and then the delegating user creates one or more temporary delegation roles and assigns the delegated users to particular roles.

ABDM and RDM2000 are closely related to our model. ABDM is an attribute based delegation model with an extended delegation constraint. The delegation constraint in ABDM includes delegation attribute expression (DAE) and delegation prerequisite conditions. The delegated user must match the delegation constraint when assigned to a

delegated role. This delegation constraint allows the delegating user to restrict the candidates for the delegated attributes more strictly. Our work borrows the role assignment mechanisms in ABDM to support role-based simultaneous delegation.

RDM2000 supports regular role delegation in a role hierarchy and sub-delegation. A rule-based specification language has been presented to enforce authorization of delegation and revocation based on RDM2000. It uses the *can_delegate* condition with prerequisite roles to restrict the scope of delegation. Our framework is based on the RDM2000 model.

1.4 Thesis Contributions

In this thesis we focus on the analysis of RDM2000 and a proposal of a delegation model called URDM. This model is an extension of RDM2000, which supports role-hierarchy, simultaneous delegation and single-step delegation.

We have achieved the following:

- We have shown a new simultaneous delegation relation in URDM which supports delegation processes simultaneously.

- We have identified URDM for role delegation while delegated users are within the same role simultaneously.

- We have explored URDM for permissions delegation while delegated users are within the same role simultaneously.

- We have demonstrated URDM for role delegation while delegated users are within different roles simultaneously.

- We have demonstrated URDM for permissions delegation while delegated users are within different roles simultaneously.

- We have demonstrated the implementation of a web application for URDM.

Our research can be of interest to the following group of people:

- Software security engineers who are developing various role-based access control models in large-scale distributed environments.

1.5 Thesis Outline

The thesis will begin with a background literature review and motivation in Chapter 2. Three areas related to our research are surveyed, including techniques of access control, common characteristics of delegation and various delegation models.

In Chapter 3, we will present the framework of URDM, including the architecture of URDM, a basic simultaneous delegation relation.

In Chapter 4, we will show simultaneous delegation in URDM in more detail.

In Chapter 5, we will address revocation in URDM.

In Chapter 6, we will demonstrate the implementation of URDM.

We will conclude this thesis and discuss the directions for future work in Chapter 7.

1.6 Summary

The protection of data against unauthorized access is a very important part of information security. Access control is the management of admission to system and network resources. Role-based access control (RBAC) is an important approach in access control in which permissions are associated with roles and users are assigned to appropriate roles.

Our research work focuses on one aspect of RBAC called delegation and has developed a simultaneous delegation model, namely URDM. URDM supports role-hierarchy, simultaneous delegation and single-step delegation.

In this chapter, we have briefly introduced our thesis work, related research work, thesis contributions, and thesis outline. Thesis background and motivation will be presented in detail in the next chapter.

Chapter 2 Background and Motivation

2.1 Introduction

Authorization is concerned with the ways in which users can access resources in the computer systems in information technology. Access control is the most fundamental and most pervasive security mechanism in use. As one aspect of a comprehensive access control solution, role-based access control (RBAC) is policy-neutral and allows for scalable collaboration while ensuring least privilege, separation of duties and other constraints. The benefit of RBAC to system administrators is the ability to assign permissions to dynamic populations of users based on their roles. Consequently, RBAC provides a mechanism for reducing the cost, complexity and the potential for access control errors. Further discussion and justification of the benefits of RBAC can be found in [Ferraiolo, et al., 2003]. The family of role-based delegation models provides a security technology for implementing RBAC in a distributed environment with authorization of users.

In this chapter, we will discuss the following areas that relate to this thesis:

- Information Security
- Access Control Models and Technologies
- Role-Based Access Control
- Delegation in Role-Based Access Control
- Characteristics of Delegation

- Role-Based Access Control Model (RBAC96)
- Role-Based Delegation Model (RBDM0)
- Role-Based Delegation Model 2000 (RDM2000)
- Permission-based Delegation Model (PBDM)
- Attribute-Based Delegation Model (ABDM)
- Motivation for Our Proposal

2.2 Information Security

Information security incidents are rising at an alarming rate every year. We can realize this major issue from Figure 2-1 [Leidigh 2005]. As the complexity of the threats increases, so do the security measures required to protect networks. Data center operators, network administrators, and other data center professionals need to comprehend the basics of security in order to safely deploy and manage networks today.

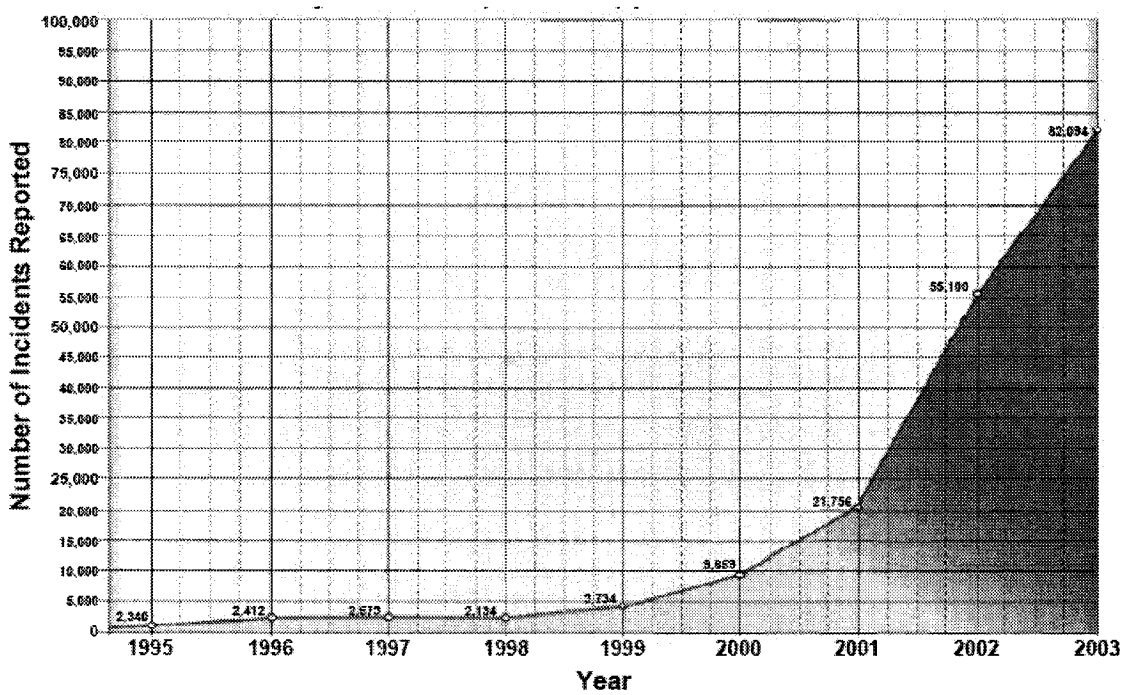

Figure 2-1. Security Incidents from 1995 to 2003

The requirements of information security within an organization have undergone two major changes in the last several decades. Before the widespread use of data processing equipment, the security of information felt to be valuable to an organization was provided primarily by physical and administrative means. An example of the former is the use of rugged filing cabinets with a combination lock for storing sensitive documents. An example of the latter is personnel screening procedures used during the hiring process.

Although it can be important to protect against all types of attacks, security does not come cheaply. Users must do a proper risk analysis to find out what the most significant sources of attack are and devote the most resources to protecting against them. Most risk analysis assessments can be divided into three categories: confidentiality, integrity, and availability.

- Confidentiality refers to limiting information access and disclosure to authorized users ("the right people"), and preventing access by or disclosure to unauthorized ones ("the wrong people"). Underpinning the goal of confidentiality are authentication methods like user-IDs and passwords, which uniquely identify a data system's users, and supporting access control methods that limit each identified user's access to the data system's resources.

- Integrity refers to the trustworthiness of information resources. It includes the concept of "data integrity", which means data has not been changed inappropriately, whether by accident or deliberately malign activity. Integrity also includes "origin" or "source integrity", which means data actually came from the person or entity that claims to have sent it, rather than an imposter. Integrity can even include the notion that the person or entity in question entered the right information. That is, information reflected the actual circumstances (in statistics, this is the concept of "validity") and that under the same circumstances would generate identical data (what statisticians call "reliability").

- Availability refers to the availability of information resources. An information system that is not available when you need it is at least as bad as none at all. It may be much worse, depending on how reliant the organization has become on a functioning computer and communications infrastructure.

As one of the most visible computer security solutions, access control is critical to preserving the confidentiality and integrity of information. The condition of confidentiality requires that only authorized users are permitted to access to information,

and the condition of integrity requires that only authorized users are permitted to alter information in authorized ways. Access control is less obviously central to preserving availability, but it clearly has an important effect: an attacker who gains unauthorized access to a system is likely to have little trouble bringing it down. Let's review the access control techniques in more detail.

2.3 Access Control

Access control is the process by which users are identified and granted certain privileges to information, systems, or resources. Controlling how network resources are accessed is paramount to protecting private and confidential information from unauthorized users. The types of access control mechanisms available for information technology initiatives today continues to increase at a breakneck pace. Most access control methodologies are based on the same underlying principles. If users can understand the underlying concepts and principles, they can apply this understanding to new products and technologies and shorten the learning curve so they can keep pace with new technology initiatives.

Access control devices properly identify people, and verify their identity through an authentication process so they can be held accountable for their actions. Good access control systems record and timestamp all communications and transactions so that access to systems and information can be audited at later dates.

Reputable access control systems all provide authentication, authorization, and administration. Authentication is a process in which users are challenged for identity credentials so that it is possible to verify that they are who they say they are. Once a user has been authenticated, authorization determines what resources a user is allowed to access. A user can be authenticated to a network domain, but only be authorized to access one system or file within that domain. Administration refers to the ability to add, delete, and modify user accounts and user account privileges.

2.3.1 Overview of Different Models of Access Control

We divide existing models of access control into two categories: traditional existing models and new developing ones. Traditional access control models include discretionary access control (DAC), mandatory access control (MAC), rule-based access control (RUAC) and role-based access control (RBAC).

- **DAC**. This type of control is used to restrict a user's access to protected objects on the system. The user may also be restricted to a subset of the possible access types available for those protected objects. Access types are the operations a user may perform on a particular object (e.g., read, write, execute). Typically, for each object, a particular user or set of users has the authority to distribute and revoke access to that object. Users may grant or rescind access to the objects they control based on "need to know" or "whom do I like" or other rules. DAC mechanisms control access based entirely on the identities of users and objects. The identity of the users and objects is the key to DAC [Sandhu 1996b]. This concept is relatively straightforward and is used in the access control matrix. An access control matrix is an array containing one row per subject in the system and one column per object. The entries in the matrix specify the operations of, or the type of access that each subject has to, each object. For many enterprises within industry and civilian government, end users do not "own" the information to which they are allowed access as is assumed by DAC policies. For these organizations, the corporation or agency is the actual "owner" of system objects, and it may not be appropriate to allow users to give away access rights to the objects.

Four limitations of DAC:

❖ DAC lets users decide the access control policies on their data, regardless of whether those policies are consistent with the global policies. Therefore, if there is a global policy, DAC has trouble ensuring consistency.

- ❖ Information can be copied from one object to another, so access to a copy is possible even if the owner of the original does not provide access to the original. This has been a major concern for military environments.

- ❖ Similarly to the previous item, flawed software can be "instructed" by attackers to change its DAC policies.

- **MAC**. MAC is defined as "A system of access control that assigns security labels or classifications to system resources and allows access only to entities (people, processes, devices) with distinct levels of authorization or clearance. These controls are enforced by the operating system or security kernel." [Computing Dictionary 2006] MAC does not allow the creator of the information to govern who can access it or modify data. Administrators and overseeing authorities pre-determine who can access and modify data, systems, and resources. MAC is most commonly applicable to Classified National Security Information where best effort mechanisms are inadequate; absloute enforcement is mandated. An NSA research project called SELinux (Security-Enhanced Linux) added mandatory access control architecture to the Linux kernel.

One limitation of MAC:

- ❖ MAC enforces access control on the basis of information security labels attached to users and objects. It shows an access control relationship that cannot be changed by the object owner. MAC can consistently determine all kinds of access controls between subjects and objects. Security labels have to be granted to all subjects and objects by the system supervisor, and can be changed only in accordance with the content of the object. Nowadays, MAC is mostly applied in military and government.

- **RUAC.** RUAC allows users to access systems and information based on predetermined and configured rules. Rules can be established that allow access to all end-users coming from a particular domain, host, network, or IP addresses. If

an employee changes their role within the organization, their existing authentication credentials remain in effect and do not need to be re-configured. Using rules in conjunction with roles adds greater flexibility because rules can be applied to people, as well as devices. "A rule makes it possible to give one or more access rights to a whole group of users making rules quite a powerful and dynamic administration tool"[Access Control Methodologies 2006]. Changes in user attributes automatically change the user's access rights. It is important to note, however, that rules can become quite unwieldy (i.e., numerous and complex) and so care must be taken to ensure that the administrative burden of RUAC does not become too great.

Three Limitations of RUAC:

- When using rules, the assignment of access permissions to users is highly dynamic. Thus, it is difficult to obtain an overview concerning who is allowed to do what.

- As result, it is not easy to maintain rules in the long run because it is difficult to foresee the impact of rule changes.

- Rules completely lack the capacity of roles to build "business roles" which contain all permissions for a specific business function and can be assigned by non-technical administrators.

- **RBAC**. RBAC allows users to access systems and information based on their role within the organization. In the RBAC models, permissions are not directly assigned to users but are instead collected in roles. Every role represents a special set of permissions. A user is assigned to one or more roles, thereby acquiring permissions defined for the roles. To facilitate the administration of permissions, roles can be organized in role hierarchies in which permissions are inherited. Permissions can easily be taken away or added to a role if necessary. This is then reflected in the permissions granted to all users that are assigned directly or

indirectly to the role. The standard RBAC model also contains sessions. During a session, a user can activate one or more of the assigned roles. Each session is associated with one user, whereas a user can have several sessions at the same time. Our work will focus on one aspect of RBAC in this thesis. Many information technology vendors have incorporated RBAC into their product line, and the technology is finding applications in areas ranging from health care to defense, in addition to the mainstream commerce systems for which it was designed.

Two Limitations of RBAC:

- The administrative issues of large systems still exist in RBAC: memberships, role inheritance, and the need for fine-grained customized privileges make administration potentially unwieldy.
- While RBAC supports data abstraction through transactions, it cannot be used to ensure permissions on sequences of operations that need to be controlled. [Sandhu 1996c]

Based on traditional access control models, several new access control models have been proposed, such as team-based access control (TMAC), task-based access control (TBAC), and coalition-based access control (CBAC).

- **TMAC**. TMAC is based on RBAC. In TMAC, users are assigned to teams and by virtue of team membership, get access to a team's resources. However, for each user, the exact permissions he/she obtains to a team's resources will be determined by his/her role and the current activity of the team. TMAC recognized the importance of context information associated with collaborative tasks and the ability to apply this context to decisions regarding permission activation. The collaboration context of a team contains two pieces: the user context, which could be the current members (users) of a team, and the object context, which could be the set of object instances required by the team to

accomplish its task. TMAC allows us to create a general structure (class/definition) of a team with role-based permission assignments to object-types. However, when a team is instantiated, the user context can be used to tailor the role-based permissions defined on object types to user-specific permissions on individual object instances considered to be part of a team's resources.

One limitation of TMAC:

- ❖ TMAC does not explain how it incorporates the team concept into a general RBAC framework. [Alotaiby and Chen 2004]

- **TBAC**. TBAC is an active approach whereby permissions are controlled and managed in such a way that they are turned-on only in a just-in time fashion and synchronized with the processing of authorizations in progressing tasks [Thomas and Sandhu 1994]. An authorization-step is a fundamental abstraction in TBAC and is used to group and manage a set of related permissions. TBAC supports the notion of a lifecycle for an authorization-step. Further, TBAC keeps track of the usage and consumption of permissions, thereby preventing the abuse of permissions through unnecessary and malicious operations. TBAC provides for the modeling of enterprise-oriented authorization policies using dependencies that relate authorizations according to some enterprise policy. Thomas and Sandhu [29] proposed a family of TBAC models: $TBAC_0$, $TBAC_1$, $TBAC_2$ and $TBAC_3$. $TBAC_0$ provides some basic facilities to model tasks, authorization-steps and dependencies relating various authorization-steps. $TBAC_1$ and $TBAC_2$ include (inherit) $TBAC_0$ but add more features. $TBAC_1$ incorporates the notion of composite authorizations whereas $TBAC_2$ adds constraints. $TBAC_3$ includes $TBAC_1$ and $TBAC_2$ and by transitivity $TBAC_0$.

One limitation of TBAC:

- ❖ The consolidated model $TBAC_3$ needs further examination. In particular, the interaction of composite-authorizations from $TBAC_1$ and constraints from $TBAC_2$ requires further study. [Thomas and Sandhu 1997]

- **CBAC**. CBAC [30] refers to a family of coalition access controls. The basic CBAC model layers coalition access control concepts on top of a simple RBAC model. The other CBAC models incorporate elements of TMAC and TBAC. These models support the use of system context information in decisions to activate, synchronize and deactivate permissions.

One limitation of CBAC:

- ❖ An administrative model for CBAC that supports both hierarchical and distributed delegation of authority hasn't been developed. Such administrative conditions are critical to the effectiveness of information resource sharing within coalitions.[Cohen et al. 2002]

2.4 Overview of Access Control Technologies

In the network security field, access control is the ability to limit and control the access to systems and applications via communication pathways. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual. Recently, various access control technologies have been used to solve enterprise access issues, including biometric devices, tokens, smart cards, and passwords.

Biometric devices authenticate users to access control systems through biological identifications of users, such as a fingerprint, voiceprint, iris scan, retina scan, facial scan, or signature dynamics. Currently, biometric-based authentication can be used for workstation, network, and domain access, single sign-on, application logon, data

protection, remote access to resources, transaction security and web security. The advantage of using biometrics is that end-users do not lose or misplace their personal identifier. It's hard to leave user's fingers at home. One example is illustrated in Figure 2-2. [Spence 2003]

Figure 2-2. An Example of Biometric Devices

Smart cards are plastic cards that embed microprocessors and memory chips. Unlike magnetic stripe cards, smart cards can carry all necessary functions and information on the cards. Therefore, they do not require access to remote databases at the time of the transaction. There are different types of smart cards: memory cards, processor cards, electronic purse cards, security cards, and JavaCards. Used to authenticate users to domains, systems, and networks, smart cards offer two-factor authentication — something a user has, and something a user knows. The card is what the user has, and the personal identification number is what the person knows. The applications of smart cards include their use as credit or ATM cards, SIMs for mobile phones, authorization cards for pay television, high-security identification and access-control cards, and public transport and public phone payment cards.

We may see one example of architecture of smart cards in Figure 2-3. [Smartcard 2006]

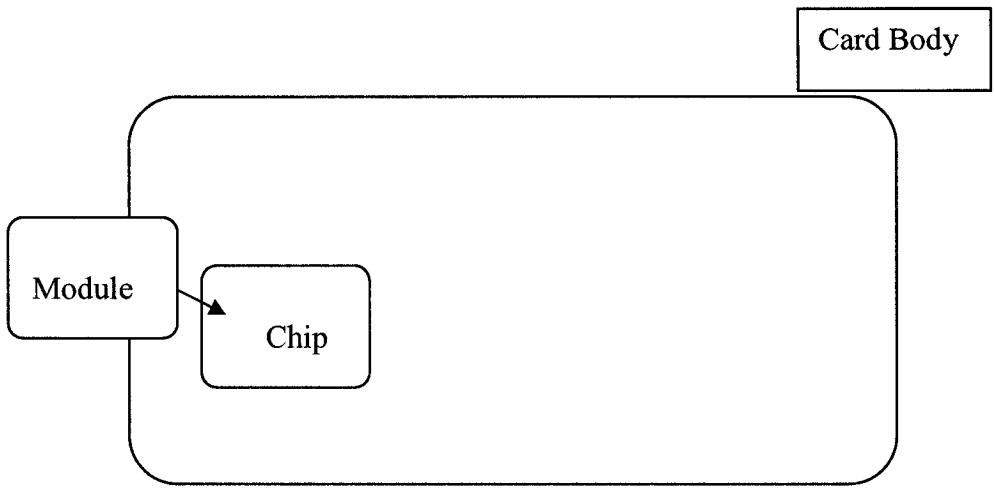

Figure 2-3. Architecture of a Smart Card

A token is a device that has a built-in challenge response method that authenticates with an enterprise server. This device displays a constantly changing ID code. A user first enters a password and then the card displays an ID that can be used to log into a network. The IDs change every 2 minutes or so. Like smart cards, tokens use two-factor authentication. However, unlike smart cards, the two-factor authentication is constantly changing based on timed intervals — therefore, when a password is entered, it cannot be reused, even if someone sniffing the wire detected it in transit. In the Windows NT architecture, a token is a system object representing the subject in access control operations, i.e. the active part (the passive part being the object being accessed). Token objects are usually built by the logon service to represent the security information known about an authenticated user, but they are essentially free-form and can include any combination and number of the possible elements.

A password is a string of characters users give to verify identification when users log in to a computer system [Password 2006]. Passwords have been widely used for access control more than any other type of solution, due to their convenience and practicality for service providers and end-users. On most systems, a password is between 6 and 8 characters long. On information technology systems, passwords can be used to write-protect documents, files, directories, and to allow access to systems and resources.

However, it is a big problem in computer security that human chosen passwords are generally insecure since a large fraction of the users choose passwords that come from a small domain. A small password domain enables adversaries to attempt to login to accounts by trying all possible passwords, until they find the correct one. This attack is referred to as a dictionary attack. If passwords are used, it is recommended that mixed-case passwords with both numeric and alphabetic characters are used, since these types of passwords are more difficult for password cracking tools to crack. Passwords with names and real words in them are easiest to crack. Good password choices look like these:

- 2cjlO0t5
- B2d5g1Tx
- 4cVDrWL8

Poor password choices look like these:

- JackTim
- Gofishing
- HappyBirthday

Because our research focuses on RBAC, the following section will describe RBAC in more detail.

2.5 Role-Based Access Control

RBAC is an approach to restricting system access to authorized users. Users, roles and permissions are three basic elements in RBAC. Within an organization, access decisions are based on an individual's roles and responsibilities within the organization or user base. Roles are created for various job functions. The process of defining roles is usually based on analyzing the fundamental goals and structure of an organization and is usually linked to the security policy. For instance, in a university organization, the different roles of users may include those such as director, professor, administrative staff, students, technology staff, etc. These members require different levels of access in order to

perform their functions, but also the types of web transactions and their allowed context vary greatly depending on the security policy and any relevant regulations. (see Figure 2-4, [Chen and Sandhu 1996])

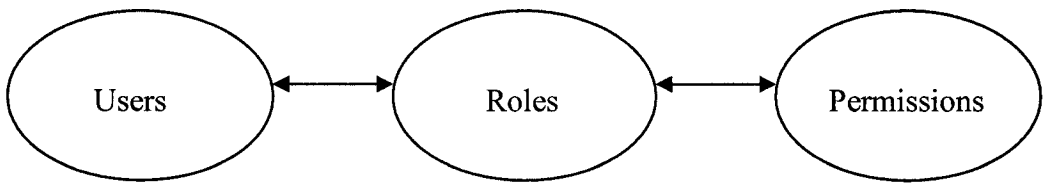

Figure 2-4. Users, Roles and Permissions Relationships

The permissions to perform certain operations are assigned to specific roles. Members of staff (or other system users) are assigned particular roles, and through those role assignments acquire the permissions to perform particular system functions. RBAC differs from access control lists (ACL's) used in traditional DAC systems in that it assigns permissions to specific operations with meaning in the organization, rather than to low level data objects. An ACL is a column of an access control matrix. An access control list could be used to permit or deny write access to a particular system file, but it would not say in what ways that file could be changed. The assignment of permission to perform a particular operation is meaningful, because the operations are fine grained and they have meanings within the application.

An RBAC framework should provide security administrators with the ability to determine who can perform what actions, when, from where, in what order, and in some cases under what related circumstances. The following aspects exhibit some of the main features of RBAC for an access control model.

- Roles are assigned based on system architecture with emphasis on the organizational security policy.
- Roles are assigned by the administrative officer based on relative relationships within the organization or user base. For instance, a manager would have certain

authorized transactions over his employees. An administrator would have certain authorized transactions over his specific realm of duties.

- Each role is designated a profile that includes all authorized commands, transactions, and allowable information access.
- Roles are granted permissions based on the principle of least privilege.
- Roles are determined with a separation of duties.
- Roles are activated statically and dynamically as appropriate to certain triggers.
- Roles can be only be transferred or delegated using strict sign-offs and procedures.
- Roles are managed centrally by a security administrator or project leader.

2.6 Delegation in Role-Based Access Control

Delegation is defined as "A person or group of persons officially elected or appointed to represent another or others."[Answers 2006] Delegation of authority is an important business rule related to access control policies. "Many motivational theories point to the importance of accountability and responsibility in determining employee behavior. Delegation can be used to support accountability and responsibility."[Stockley 2006] There exist three kinds of situations in which delegation is required:

- Temporary role. When an individual is absent because of a business trip or some other reasons, the job operations need to be maintained by others. This requires that somebody be delegated the authority to do the absent individual's job.
- Distribution requirements. When a system is initially set up or will soon be reorganized, job functions are distributed from higher job positions to lower job positions in the organizational structure.
- Information sharing. In the same system or organization, because of task requirements, people need to get some access authority to share resources.

Various definitions of delegation have been proposed in the literature. Most commonly delegation has been studied as user-to-machine, machine-to-machine and human-to-

human delegation [Barka]. Gasser and McDermott [Gasser, Morrie and McDermott 1990] 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 his 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 of those for which the user is authorized.

Gladny [Gladny 1997] 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 proposed 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. Scaling to many users is accomplished by emulating vertical delegation in organizational hierarchies, extended to permit privilege delegation from any node to any other node, up, down, or across the organization tree; this provided 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.

Recently, user-to-user delegation has been considered. In user-to-user delegation, there are five components: a delegating user, a delegating role, a delegated user, a delegated role, and associated constraints. 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. Thus, the delegating user handles this delegation process by himself or herself; no additional administrative officer is involved in such a delegation. RDM2000, ABDM and PBDM are three models of user-to-user delegation.

2.7 Characteristics of Delegation

Ezedin Barka and Ravi Sandhu [Barka and Sandhu 2002] addressed definitions of delegation characteristics. These include eight features: *permanence, monotonicity, totality, administration, levels of delegation, multiple delegation, agreements,* and *revocation.* These characteristics are listed as below:

- *Permanence* means 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. Temporary delegation means that delegation is limited by time.

- *Monotonicity* refers to the state of the authorization that the delegating user owns after this user delegates the role. The delegating role member maintains the power of his or her role when a monotonic delegation is called. If the delegating user loses the power of the delegated role for the duration of delegation, it can be termed a non-monotonic delegation.

- *Totality* addresses how completely the permissions assigned to the role are delegated. Two options are discussed: total delegation and partial delegation.

- *Administration* describes the actual administrator of the delegation. Two types of administration for delegation are defined. One is called self-acted delegation, in which the delegating user manages the delegation himself; the other is named agent-acted delegation, which means the delegating user empowers a third party to manage the delegation on the user's behalf.

- *Levels of delegation* determine whether the delegation can be further delegated and how many times the delegation can be delegated. If it refers to multiple-step delegation, it is called *sub-delegation* in this thesis.

- *Multiple delegation* focuses on the number of people to whom a delegating user can delegate at any given time. It is called *simultaneous delegation* in this thesis.

- *Agreements* define the delegation protocol between the delegating user and the delegated user. Two types are proposed: bilateral agreement and unilateral agreement. A bilateral agreement is an agreement that delegation is accepted by both the delegating user and the delegated user. A unilateral agreement is an agreement that only the delegating user can determine to delegate the role. Once the delegating user confirms the delegated user, the delegated user has to accept the responsibility.

- *Revocation* is a process that accompanies delegation: a delegating user can remove the rights that the delegating user delegated to a delegated user who is a member of another role.

Based on above characteristics, various delegation models have been proposed. The following sections will give an overview of these models which are related to our thesis work.

2.8 Role-Based Access Control Model (RBAC96)

RBAC96 is a user-to-user delegation model. It concentrates on the ability of a user who belongs to one role to delegate his role to another user who belongs to another role. Roles can obtain new permissions. Any permission can be revoked from a role if necessary. The RBAC96 model is based on five sets of entities called Users (*U*), Roles (*R*), Permissions (*P*), Sessions (*S*), and Constraints (see Figure 2-5) [Chou 2004].

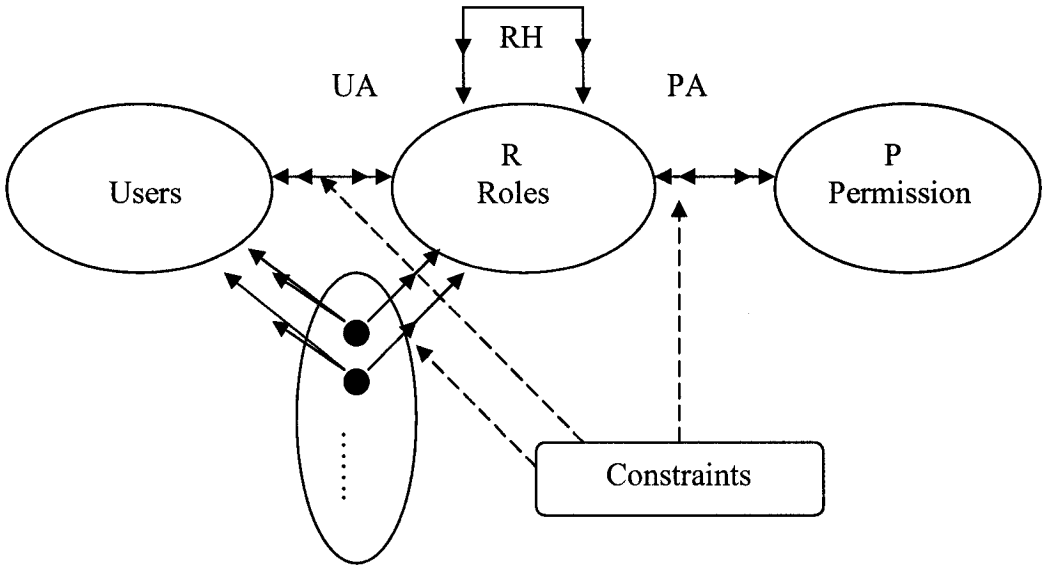

Figure 2-5. RBAC96 Model

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. A session (*S*) is a mapping relationship between a user and a set of activated roles. A constraint can be associated with the user-role assignment or with the activation of roles within user sessions.

The definition for the RBAC96 model can be summarized as below:

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

2. UA $\subseteq$ U $\times$ R, which is a many-to-many user assignment relation assigning user to roles.

3. PA $\subseteq$ P $\times$ R, which is a many-to-many permission assignment relation assigning permissions to roles.

4. RH $\subseteq$ P $\times$ R is a partial order on R called role hierarchy.

5. User: $S \rightarrow U$, is a function mapping each session S, to the single user $user(s_i)$ and is constant for the session's lifetime.

6. Roles: $S \rightarrow 2^R$ is a function that maps each session S_i to a set of roles, where $Role(S_i) \subseteq \{r \mid (\exists r \leq r')[(user(s_i), r') \in UA]\}$.

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

8. Permissions: $S \rightarrow 2^P$ is a function derived from PA mapping each session s_i to a set of permissions where $Permissions(s_i) = \{p \mid [(\exists r \leq r'), (p, r') \in PA, r' \in Role(s_i)]\}$.

2.9 Role-Based Delegation Model (RBDM0)

RBDM0 is the simplest version of the RBDM models in which inheritance of permissions between roles and constraints are not considered. In RBDM0, two types of members of delegating role *r* are defined as below [Barka and Sandhu 2002b]:

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

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

RBDM0 is also defined by several assumptions:

- Delegation cannot be activated between members in the same role.
- Only one-step delegation is addressed.
- The delegation is total.
- Any original member in a role can revoke the delegation to any delegated member in that role.
- Duration (T) is a constraint of time associated with each unit of delegation.

Several new features are added to the above RBAC96 model to build 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.
- $Users\_O(r) = \{U \mid (U, r) \in UAO\}$.
- $Users\_D(r) = \{U \mid (U, r) \in UAD\}$.
- 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.
- RBDM0 controls user-user delegation by means of the relation *can_delegate* $\subseteq R \times R$.
- Revocation is activated by timeout.
- Revocation is activated by role members.

2.10 Role-Based Delegation Model (RDM2000)

RDM2000 is a user-to-user delegation model which supports role hierarchy and sub-delegation. This model is based on RBAC96 and RBDM0. Two assumptions are addressed: each user-role assignment is unique, so that a delegation can be identified by a unique user role assignment by delegated users; a user who is a member of role *r* (whether it is explicit or implicit membership) cannot be delegated the same role *r*, thus preventing cycles in delegations. RDM2000 defines a new delegation relation *DLGT*. This relation contains sets of three elements, namely, original user assignments, delegated user assignments and constraints, and is a one-to-many relationship on user assignments. *DLGT* includes two delegation relations, namely, an original user delegation relation *ODLGT* and a delegated user delegation relation *DDLGT*. *DLGT* also supports partial delegation in a role hierarchy. RDM2000 supports sub-delegation by introducing the concept of maximum delegation depth. Delegation depth is defined as the number of times each delegation can be further delegated. A delegation tree is presented in RDM2000 which is composed of various delegation paths in a hierarchical structure. Delegation path is an ordered list of user assignment relations generated through sub-delegation. This delegation path begins from an original user assignment. If various delegation paths start at the same original user assignment, these paths create one delegation tree. Before the delegation process, RDM2000 also defines a relation called *can_delegate* and cites a prerequisite condition to restrict the delegation candidates. This relation is used to express the scenario "a user who is a member of role can delegate the role to any user whose current entitlements in roles satisfy the prerequisite condition without exceeding the maximum delegation depth." [Zhang et al. 2001] A prerequisite condition *CR* is adopted from ARBAC97 [Sandhu 1999].

Revocation is the reverse process of delegation. RDM2000 supports multiple revocation types, such as Grant-Dependent Revocation, Grant-Independent Revocation; Strong Revocation and Weak Revocation; Cascading Revocation and Non-Cascading revocation.

2.11 Permission-based Delegation Model (PBDM)

In many cases, the delegating user wants to delegate only some permissions instead of all permissions which are associated with the role. Permission-based Delegation Model (PBDM) is proposed to handle this kind of situation. PBDM supports user-to-user delegation and role-to-role delegation. PBDM is composed of a set of delegation models: $PBDM_0$, $PBDM_1$ and $PBDM_2$ [Zhang et al. 2003].

- $PBDM_0$ is designed as a user-to-user delegation model. It supports single-step delegation, sub-delegation and temporary delegation. In $PBDM_0$, the delegating user can delegate both permissions and roles to the delegated users. Roles are partitioned into regular roles (*RR*) and delegation roles (*DTR*). This partition induces a parallel partition of *UA* and *PA*. *UA* is separated into user-regular role assignment (*UAR*) and user-delegation role assignment (*UAD*). *PA* is similarly separated into permission-regular role assignment (*PAR*) and permission-delegation role assignment (*PAD*). Delegation role can be placed in the regular role hierarchy when a delegating user delegates a regular role or roles to a delegated user; otherwise it is isolated from the hierarchy.

- $PBDM_1$ includes three different layers of roles: regular roles (*RR*), delegatable roles (*DBR*), and delegation roles (*DTR*). Permissions assigned to regular roles cannot be delegated to other roles or users. A delegatable role is the role whose permission can be delegated to other roles or users by creating delegation role. For each delegatable role there is a regular role on which it is based. There is a one-to-one mapping between regular roles and delegatable roles. The users that assigned to a delegatable role are exactly the same as those assigned to the regular role it is based on. In $PBDM_1$, each delegatable role is created by a security administrator. The permission-delegatable role assignment (*PAB*) is also managed by a security administrator. Therefore the security administrator can control the permission flow by assigning different permissions to a delegatable role. Security administrators manage the permission-regular role assignment (*PAR*), user-regular

role assignment (*UAR*), permission-delegatable role assignment (*PAB*), and user-delegatable role assignment (*UAB*). Individual users manage permission-delegation role assignment (*PAD*) and user-delegation role assignment (*UAD*).

- $PBDM_2$ is a role-to-role delegation model. This means all the delegation authority will be managed by a trusted third party, namely, a security officer. An individual user cannot own any delegation roles and permissions. This point is different from user-to-user delegation. There are four different layers in $PBDM_2$: regular roles (*RR*), fixed delegatable roles (*FDBR*), temporal delegatable roles (*TDBR*), and delegation roles (*DTR*). A delegating user is a fixed delegatable role. A temporal delegatable role has the permissions that it receives from the delegating user with role-to-role assignment. There is a one-to-one mapping between RR, FDBR and TDBR. The users that are assigned to a temporal delegatable role, a fixed delegatable role and a regular role are exactly the same if these three roles are in the one-to-one relation. This means that a pair of (regular role, fixed delegatable role, temporal delegatable role) will be used as a single role in user-role assignment. By dividing the delegatable role into fixed and temporal delegatable roles, a temporal role can receive permissions delegated by a fixed delegatable role. Since there is no role hierarchy for TDBR, invalid permission flow will not happen, and a role-to-role delegation can be achieved.

2.12 Attribute-Based Delegation Model (ABDM)

Within a system, a few positions and users in the same position may have different qualifications and abilities. For RBAC purposes, it's impossible to create different roles for different users with different qualifications and abilities, because this will increase the total number of roles remarkably.

Attribute-Based Delegation Model (ABDM) is proposed to handle this problem. The core contribution of this model is that ABDM develops its delegation constraint. This constraint contains delegation prerequisite condition (CR) and delegation attribute

expression (DAE). Only those delegating users whose prerequisite roles and DAE satisfy CR and DAE of delegation constraint can be assigned to a delegation role. In ABDM, DAE and CR form a strict delegation constraint in delegation. We will borrow this delegation constraint from ABDM in our proposal.

2.13 Thesis Motivation

In a distributed environment, a delegating user may need to delegate his or her role to more than one person at the same time. It is also possible that a delegating user wants to delegate only a set of permissions to more than one candidate simultaneously. That is, a delegated role with a full set of permissions or a subset of permissions will be delegated in this simultaneous delegation process. For example, a professor may delegate the role of research assistant to some members of his/her research group at the same time, or only delegate his/her access right to a new established computer lab to some members of his/her research group at the same time. This type of delegation is referred to as the simultaneous delegation problem. Existing delegation models, such as RDM2000, PBDM and ABDM only handle the delegation process one by one. If simultaneous delegation is required in a large-scale environment, these models cannot work efficiently. The goal of this thesis is to propose a new delegation model (expressivity) and make delegation processes more efficient (performance). Expressivity means the quality of a written manifestation to show with full feeling or meaning the ideas of the author; that is, the ability to express some concept(s) that the author wishes to express. In this thesis, expressivity means that users can give permissions to a role, without knowing in advance which individual(s) might occupy that role tomorrow. A role-based simultaneous delegation model called User-to-Role Delegation Model (URDM) will be represented in this thesis. URDM is an extension of RDM2000, which supports simultaneous delegation, single-step delegation, and role hierarchy.

2.14 Summary

Access control is one of the most important security technologies in information systems. As an alternative to DAC and MAC, RBAC security technology has gained considerable attention recently, and become the predominant model for advanced access control because it reduces the complexity and cost of security administration in large networked applications.

In this chapter, we have reviewed information security, access control models and related technologies. Especially, we gave an overview of RBAC. Delegation is an important part of RBAC. After introducing the primary features of delegation, various delegation models have been briefly presented. RDM2000 is one of these models, which supports role hierarchy and sub-delegation. However, this model does not support simultaneous delegation. Our thesis is to propose a new role-based delegation model called User-to-Role Delegation Model (URDM) to support such simultaneous delegation.

Starting from the next chapter, we will show URDM in detail.

Chapter 3 Architecture of URDM

3.1 Introduction

In RBAC, a senior role inherits a junior role's permissions by virtue of the role hierarchy. But a junior role is not allowed to carry out a senior role's permissions. Various delegation models have been proposed to handle this problem, such as RDM2000. However, RDM2000 does not support simultaneous delegation.

In this chapter, we propose a new delegation model called User-to-Role Delegation Model (URDM), which is an extension of RDM2000. URDM supports simultaneous delegation, role hierarchy, and single-step delegation.

The following topics will be covered in this chapter:

- The Basic Architecture of URDM
- A New Simultaneous Delegation Relation
- Single Delegation and Single-Step Delegation

3.2 Role-based Simultaneous Delegation

Access control is what allows the certified user to access the inner information of a system within the limits of permission. RBAC means access decisions are based on roles. In RBAC, a senior role inherits a junior role's permissions by virtue of the role hierarchy. But a junior role does not have the permissions which are only granted to a senior role or to other role groups. When a senior role fails to operate, junior roles may not continue to perform their jobs when they need the senior role's permission. In order to solve this problem, various delegation models have been proposed for different purposes. RDM2000 has been proposed to support role hierarchy and sub-delegation. However, one challenge remains to be explored in RDM2000: In many cases, a delegating role member needs to delegate his or her role to more than one person at the same time. We propose a new role-based delegation model called User-to-Role Delegation Model (URDM), which supports simultaneous delegation, role hierarchy and single step delegation.

3.3 Preparation for URDM

Based on RDM2000, URDM supports role hierarchy, single-step delegation and simultaneous delegation by presenting a new delegation relation. This delegation relation makes our model more efficient in large-scale environments: RDM2000 and ABDM can handle a single delegation process one by one; URDM can achieve multiple delegations simultaneously.

3.3.1 Assumptions

Some assumptions are discussed before introducing the functional components in URDM. We consider simultaneous delegation to be the maximum delegation width, and sub-delegation to be the maximum delegation depth. The maximum delegation width can be also considered as how many separate delegation processes are operated at the same level; the maximum delegation depth can be considered as how many levels the delegation process can be operated further (see Figure 3-1). Delegation between users in

the same role is not considered, because they already share the same permissions. During a user-role assignment step, the security officers take the administrative role. In a delegation step, a delegating user is to assign the delegated role to at least one delegated user in another role simultaneously; at the same time, this delegating user manages the entire delegation process by himself/herself. In this way, a single delegation can be considered as a special case of simultaneous delegation. Because permanent delegation can create accountability problems, only temporary delegation will be addressed in our model. To simplify our model, we consider only single-step delegation in this thesis.

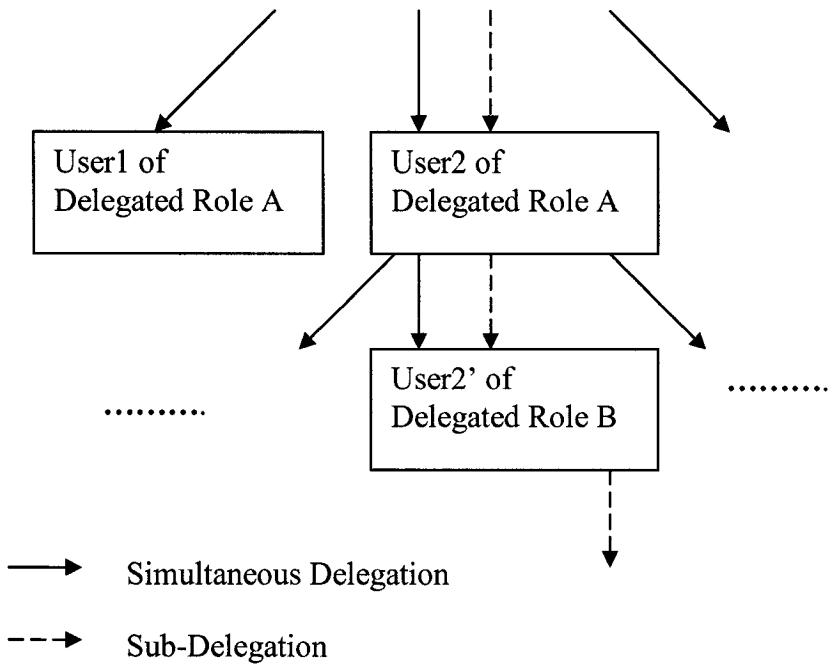

Figure 3-1. Relationship between Simultaneous Delegation and Sub-Delegation

3.3.2 An Example

Figure 3-2 illustrates an example of organizational role hierarchy and related users in a department of a university. This example will be used in the thesis to clarify our model. Figure 3-2(a) presents a role hierarchy in a school of information technology and

engineering in a university. *DIR* is the senior-most role in this academic unit. There are two academic programs in this school: Software Engineering (SE) and Electronic Engineering (EE). Each program has a senior-most PROFESSOR role (*PROF1, PROF2*) and a junior-most RESEARCH GROUP role (*RG1, RG2*). Two incompatible roles are between the above two roles: POST DOCTOAL FELLOW (*PDF1, PDF2*) and RESEARCH ASSISTANT (*RA1, RA2*). Figure 3-2(b) presents users and their role memberships. Security officers assign these users to roles before delegation.

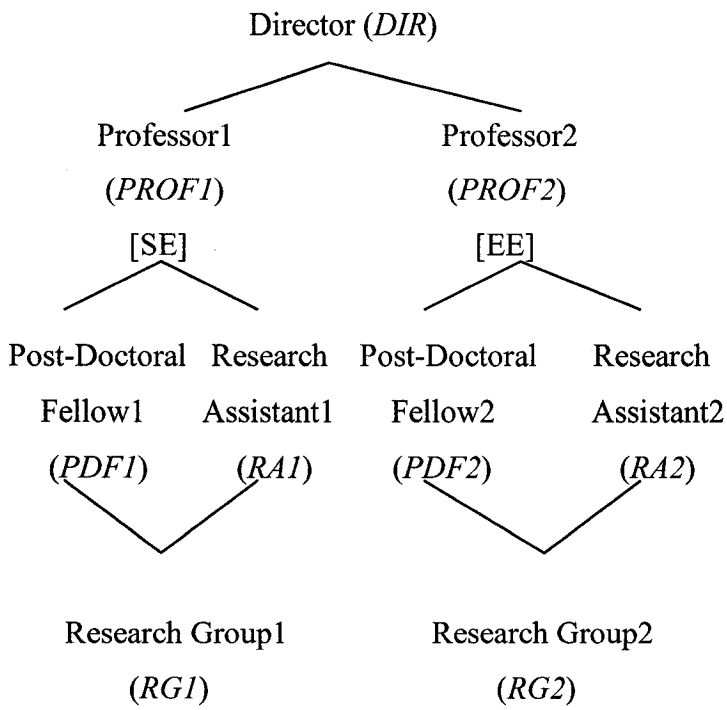

Figure 3-2 (a) Role Hierarchy

DIR: Peter

PROF1: Martin	*PROF2*: David
PDF1: Richard, Tim	*PDF2*: Mary, John
RA1: Mike, Lisa, Jack	*RA2*: Jane, Tina, Eric

Figure 3-2 (b) Users within Roles

3.4 Architecture of URDM

The framework of URDM is shown in this section. This framework includes three parts:

- Basic Elements with Functionalities from RDM2000
- Two Attribute Expressions from ABDM
- A Basic Simultaneous Delegation Relation

3.4.1 Basic Elements with Functionalities from RDM2000

We quote the following terms from RDM2000 to create the architecture of URDM. [Zhang et al. 2001]

Definition 1

- U, R, P and S are sets of users, roles, permissions, and sessions, 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. A partial order is a binary relation that is reflexive, transitive, and antisymmetric.
- Sessions: $U \rightarrow 2^S$ is a function that maps a user to a set of sessions, where 2^S is a standard set theory notation for the power set of S, the set of all subsets of the set S. Note that there are $2^{|S|}$ subsets of S; this function maps a user to one of these subsets.
- Roles: $S \rightarrow 2^R$ is a function that maps each session s_i to a set of roles, where $Role(s_i) \subseteq \{r | (\exists r' \geq r)[(user(s_i), r') \in UA]\}$
- Permissions: $S \rightarrow 2^P$ is a function derived from PA mapping each session s_i to a set of permissions where $Permission(s_i) = \{p | [(\exists r' \geq r)(p, r') \in PA, r' \in Role(s_i)]\}$
- $UAO \subseteq U \times R$ is a many-to-many original user to role assignment relation.
- $UAD \subseteq U \times R$ is a many-to-many delegated user to role assignment relation.

- UA = UAO ∪ UAD.
- Users: R → 2^U is a function mapping each role to a set of users.
  Users(r) = $\{u \mid (u, r) \in UA\}$
- Users(r) = Users_O(r) ∪ Users_D(r)
  Users_O(r) = $\{u \mid (\exists r' \geq r)(u, r') \in UAO\}$
  Users_D(r) = $\{u \mid (\exists r' \geq r)(u, r') \in UAD\}$

Role hierarchies are a natural means for structuring roles to reflect an organization's lines of authority and responsibility, and are organized in a partial order ≥, so that if $r' \geq r$ then role r' inherits the permissions of r. A member of r' is also implicitly a member of r. in such a case, r' is said to be senior to r.

3.4.2 Two Attribute Expressions from ABDM

Two constraints of delegation model ABDM are also added into URDM: Delegation prerequisite condition (PC) and Delegation attribute expression (DAE).

Only those delegated users whose prerequisite roles and attribute expression satisfy the PC and DAE delegation constraints can be assigned to a delegated role.

3.4.3 A Basic Simultaneous Delegation Relation

A new delegation relation called Simultaneous Delegation Relation (S*DR*) is proposed in this model. This relation contains two sets of components: original user assignments UAO, and delegated user assignments UAD. Five types of constraints are addressed in a pre-condition called *can_delegate* before delegation is invoked: delegation prerequisite conditions (PC), delegation attribute expressions (DAE), delegation width (DW), delegation depth (DD), and duration or delegation time (DT). We consider PC and DAE as the constraints of separation of duty; DW, DD and DT are the constraints of least privilege. PC and DAE are used to identify which candidate satisfies with the delegation requirements. Based on the DW constraint, our model can limit how many delegation jobs can be

processed simultaneously. DD refers to the maximum delegation depth. Because our model is used to consider temporary delegation, DT is the constraint that refers to temporary delegation limited by time. There are four important elements in a user-to-user delegation model: a delegating role, a delegating user, a delegated role, and a delegated user. Similarly, our URDM includes four elements as well: a delegating role, a delegating user, a delegated role, and a set of delegated users (indeed, a resulting role). A single delegated user becomes a special case of such set of delegated users.

We formalize the above description as follows:

Definition 2

- current_users CU: a set of users in the delegated role after delegation, CU $\subset$ U.
- original_users OU: a set of users originally assigned to the delegated role before delegation, OU $\subset$ U, Users_O(r) $\subset$ OU.
- delegated_users DU: a set of users delegated to the delegated role, where DU $\subset$ U, Users_D(r) $\subset$ DU.
- CU = OU $\cup$ DU.
- PC: all members of a regular role to be delegated, DU $\subseteq$ PC.
- DW: a number that refers to the maximum delegation width.
- DD: a number that refers to the maximum delegation depth.
- DAE: attribute expressions that refer to the delegation requirements for the delegated candidates.
- DT: a set of duration limitation for the delegation.
- *SDR* $\subseteq$ UA $\times$ UA is a one-to-many delegation relation. This delegation relation can be expressed by ((u, r), (DU, r')) $\in$ S*DR*.
- *can_delegate* $\subseteq$ R $\times$ PC $\times$ {DW} $\times$ {DD} $\times$ DAE $\times$ DT.

3.5 Single Delegation and Single-Step Delegation

Simultaneous delegation refers to how many separate delegation processes can be operated from the same delegation user at the same time. If the delegating user only delegates the role and/or permission to one single delegated user, this type of delegation is referred to as a single delegation, or a delegation with a delegation width of 1. Single delegation is a special case of simultaneous delegation. Sub-delegation means how deep the delegation process will go from the original delegation user. In RDM2000, sub-delegation is considered as a delegation path. If the delegating user starts to delegate the role and/or permissions to a single delegated user u_1, and there are no further delegations from u_1, we refer to such delegation as a single-step delegation, or delegation with a delegation depth of 1. Single-step delegation is a special case of sub-delegation.

If single delegation and single step delegation have a common delegating user and delegated user, then single delegation equals single-step delegation.

We formalize the above discussion and add the definitions below into URDM:

Definition 3

- SD: single delegation, the maximum delegation width is 1.
- STD: single step delegation, the maximum delegation depth is 1.
- SD = STD, if SD and STD have the same delegating user and delegated user.

3.6 Summary

Various delegation models have been proposed to support junior roles to acquire permissions from the senior roles. RDM2000 supports role hierarchy, sub-delegation, but not simultaneous delegation.

In this chapter, we have proposed a new role-based delegation model called User-to-Role Delegation Model (URDM). This model supports role hierarchy, simultaneous delegation, and single-step delegation. We have listed some assumptions and created the basic architecture of URDM. One example that will be used in the thesis to clarify our model is also given. Single delegation and single-step delegation are two special cases. We have discussed the relationship between them.

In the next chapter, we will address the simultaneous delegation process of URDM in more detail.

Chapter 4 Delegation in URDM

4.1 Introduction

URDM is designed to support simultaneous delegation processes for different purposes. There are several situations before delegation: the delegating user can delegate a role and /or permissions to a set of users simultaneously; these delegated users can be members of the same role, or they can be different members of different roles before delegation.

In this chapter, we will focus on addressing simultaneous delegation in the above situations. In order to show the delegation process systematically, a technique called Finite State Machine will be used to help formalize this process.

4.2 Assumptions

Before delegation, we need some additional assumptions as follows:

- There are four states (entities):
  --- a delegating user in a delegating role *du*;
  --- a set of regular roles $sdu_i\ i \in N$, to which a set of delegated users belong before delegation;
  --- a delegated role *DR*;
  --- a subset of permissions SP.

- URDM supports temporary delegation, single-step delegation, simultaneous delegation, monotonic delegation, total delegation, partial delegation, bilateral delegation, self-acted delegation, and revocation.

4.3 Analysis before Delegation: Four Situations

In RBAC, permissions to perform certain operations are assigned to specific roles. Members of staff (or other system users) are assigned particular roles, and through those role assignments acquire the permissions to perform particular system functions. There are four situations **before** simultaneous delegation occurs:

- *Situation 1*: The delegating user cannot delegate arbitrary subsets of permissions in *du* but can only delegate the entire role at a time to a set of delegated users, who are the members of the same *sdu*.

- *Situation 2*: The delegating user can delegate subsets of permissions in *du* at a time to a set of delegated users, who are the members of the same *sdu*.

- *Situation 3*: The delegating user can only delegate the entire role at a time to a set of delegated users, who are members of different sdu_i.

- *Situation 4*: The delegating user can delegate subsets of permissions in *du* at a time to a set of delegated users, who are members of different sdu_i.

In the following sections, we will discuss the simultaneous delegation of URDM in the above situations separately. Especially, when we review those existing delegation models, such as RDM2000 and PBDM, we find that those delegation processes have not been formalized, only some simple examples are shown. This makes the delegation process unclear for understanding. In this thesis, we clarify the delegation process in URDM by using a technique called Finite State Machine (FSM).

4.4 What is a Finite State Machine?

Finite State Machine (FSM) is defined as "A model of computation consisting of a set of states, a start state, an input alphabet, and a transition function that maps input symbols and current states to a next state. Computation begins in the start state with an input string. It changes to new states depending on the transition function. There are many variants, for instance, machines having actions (outputs) associated with transitions (Mealy machine) or states (Moore machine), multiple start states, transitions conditioned on no input symbol (a null) or more than one transition for a given symbol and state (nondeterministic finite state machine), one or more states designated as accepting states (recognizer), etc." [NIST 2006]

A FSM *M* (see Figure 4-1.) [Ural 2004] can be characterized by a quintuple $(S, I, O, \delta, \lambda)$, where

- S: finite non-empty set of states;
- I: finite non-empty set of inputs;
- O: finite non-empty set of outputs;

- $\delta : S \times I \rightarrow S$ next state function;
- $\lambda : S \times I \rightarrow O$ output function.

A state transition from state s_j to state s_k in M is denoted by

- $t_{jk} = (s_j, i/o, s_k)$ where
- s_j is the present state;
- s_k is the next state;
- i/o is the label of the transition.

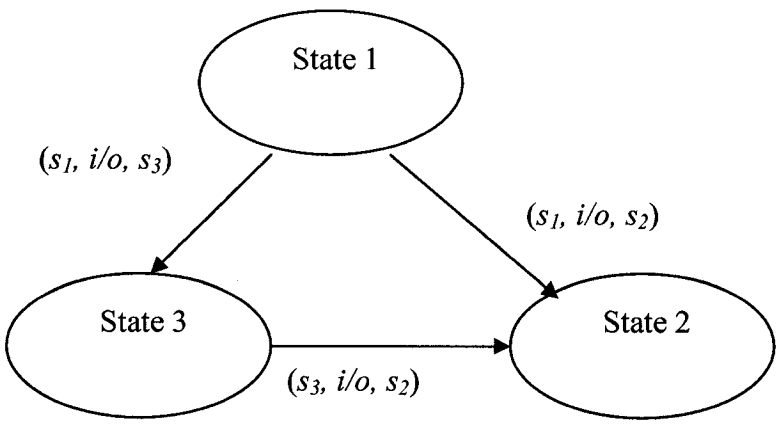

Figure 4-1: A FSM M

4.5 Delegation in *Situation1*

We add the following definitions when URDM is in *situation1*.

Definition 4

- DAE_1: a set of attribute expressions that refers to the delegation requirements for the delegated candidates within the same role *sdu* (post-conditions).
- $can\_delegate_1 \subseteq R \times PC \times \{DW\} \times \{DD\} \times DAE_1 \times DT$.

The meaning of $(r, pc, dw, dd, dae_1, dt) \in can\_delegate_1$ means a member of role r (or a role senior to r) can delegate role r (or a role junior to role r) to a set of other users who satisfy the prerequisite condition pc and dae_1 without exceeding the maximum delegation width dw, the maximum delegation depth dd, and duration constraint dt. In this thesis, dd=1. For example, if $(PROF1, RA1, 2, 1, PHD, FALL2005) \in can\_delegate_1$, then Professor Martin can delegate *PDF1* to Lisa and Mike simultaneously during the fall term 2005; Lisa and Mike are members of *RA1* before delegation, and are PHD students, so that $(Martin, PROF1, \{Lisa, Mike\}, PDF1) \in SDR$. Lisa and Mike cannot delegate the delegated role to anyone else (single-step delegation). If $dw = 1$, then URDM carries out a single delegation.

4.5.1 Delegation in FSM

In situation 1, there are three states *du*, *sdu*, and *DR* in the URDM delegation process (see Figure 4-2). When the simultaneous delegation process in *situation 1* is called, a total of six state-transitions occur:

1. (*du*, *delegation request/candidates searching*, *sdu*): a delegating user *du* sends out the delegation requests and will start to look for delegated candidates in *sdu*.

2. (*sdu, request received/searching accepted, du*): a set of delegated candidates in *sdu* received the delegation request and respond to *du* to accept the delegation process.

3. (*du, new members acceptance request/ready for acceptance, DR*): *du* sends the accept- new-members request to the delegated role *DR*, *DR* is ready to accept new members.

4. (*sdu, successful candidates selected/successful candidates added, DR*): a set of candidates have been selected from *sdu* and added into the delegated role *DR*.

5. (*DR, new members acceptance/delegation end inquire, du*): the delegated role *DR* sends the *new member acceptance* message and inquires END message of the delegation process to *du.*

6. (*du, delegation end inquire received/response delegation End message, DR*): the delegating user *du* received the request to end the delegation, and responds with the delegation END message to *DR.*

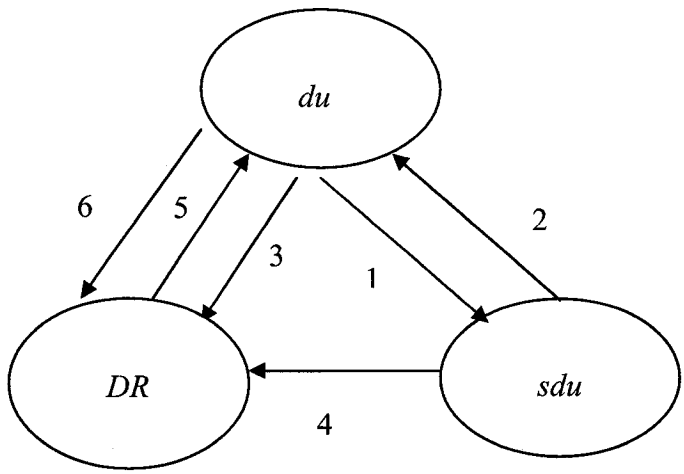

Figure 4-2. Delegation Process of URDM in FSM for Situation 1

4.5.2 A Proof of Delegation in FSM

Let s_i and s_j be two states of an FSM. States s_i and s_j are equivalent iff s_i and s_j, when excited by any sequence of input symbols, produce identical output sequences; otherwise, they are ***distinguishable*** [Ural 2004]. If every pair of its states is distinguishable, then an FSM is ***minimal***[Ural 2004]. If all states with transitions match the steps of the entire delegation process, then this FSM is ***complete.*** In this thesis, we use a FSM model of delegation in URDM and show that it is minimal and complete in order to give an accurate description of the actual protocol.

i. *Complete.* Three states are involved in *situation 1* before delegation: a delegating user in a delegating role; a regular role, whose members will be selected and participate the delegation process; a delegated role, such that successful

candidates are delegated into this role. Based on delegation requirements, a delegating user selects some members from a regular role, and delegates these members into a delegated role simultaneously. FSM in Figure 4-2 shows this process completely: transitions {1, 2, 3, 5, 6} support self-acted delegation; transition 4 supports total delegation and simultaneous delegation; transition 2 supports bilateral delegation.

ii. *Minimal.* Three states are involved in *situation 1* when delegation occurs: a delegating user, a regular role and a delegated role. Each pair of these states is distinguishable.

4.6 Delegation in *Situation 2*

It is possible that the delegating user only plans to delegate some of his/her permissions instead of all permissions which are associated with the delegated role. In order to support such simultaneous delegation in URDM, we modify the delegation relation *SDR* to *SDR'* and add the following into definition 5.

Definition 5

- SP: a subset of P, where SP $\subseteq$ P; $sp_i \in$ SP, $i \in N$.
- *SDR'* $\subseteq$ UA $\times$ UA is a one-to-many delegation relation. This delegation relation can be expressed by $((u, r), (\text{DU}, r' <\text{-} \{sp_1, sp_2, ..., sp_i\})) \in SDR'$, where $\{sp_1, sp_2, ..., sp_i\}$ is a set of permissions associated with r'.

The meaning of $(r, pc, dw, dd, dae_1, dt) \in can\_delegate_1$ is a little different from delegation in *situation 1;* that is, a member of role r (or a role senior to r) can delegate a role r with a set of permissions sp_i (or a role with a set of permissions sp_i junior to role r) to a set of other users who satisfy the prerequisite condition pc and dae_1 without exceeding the maximum delegation width dw, the maximum delegation depth dd, and the duration constraint dt. Again, dd=1. For example, if (*PROF1, RA1, 2, 1, PHD, FALL2005*) $\in$ $can\_delegate_1$, then Professor Martin can delegate *PDF1* with a subset of permissions {*lab access, conference attendance guaranteed*} to Lisa and Mike simultaneously during the fall term 2005; Mike and Lisa are members of *RA1* before

delegation, and are PHD students, so that (*Martin, PROF1, {Lisa, Mike}, PDF1<-{lab access, conference attendance guaranteed}*) ∈ *SDR'*. Lisa and Mike cannot delegate the delegated role to anyone else (single-step delegation). If *dw* = 1, then URDM carries out a single delegation. If Professor Martin delegates *PDF1* with all permissions associated with *PDF1*, this simultaneous delegation is exactly same as the delegation in *situation 1*.

4.6.1 Delegation Process in FSM

There are four states *du*, *sdu*, *DR* and *SP* in the URDM delegation process (see Figure 4-3). When the simultaneous delegation process in *situation 2* is involved, a total of nine state-transitions occur:

1. (*du, new members conditional-acceptance request/conditional acceptance request received, DR*): *du* sends the conditional-accept-new-members request to the delegated role *DR*, *DR* knows the request.

2. (*DR, delegated permissions searched/delegated permissions ready, SP*): According to *du* request, *DR* searches those permissions that will be delegated and collects them for delegation.

3. (*SP, delegated permission check request/delegated permissions matched, du*): *SP* sends the delegated permission check request to *du,* the collected permissions by *DR* match the request of *du*.

4. (*du, delegation request/candidates searching, sdu*): *du* sends out the delegation requests and will start to look for delegated candidates in *sdu*.

5. (*sdu, request received/searching accepted, du*): a set of delegated candidates in *sdu* received the delegation request and respond to *du* to accept the delegation.

6. (*sdu, successful candidates selected/ permissions delegated, SP*): a set of candidates have been selected from *sdu* and the collected permissions are delegated.

7. (*SP, permissions that have been delegated informed/permissions arrangement, DR*): *DR* knows that permissions have been delegated and arranges the current permission use list after delegation.

8. (*DR, new members acceptance/delegation end inquire, du*): the delegated role *DR* sends the *new member with conditional acceptance* message and informs *du* that the delegation process is complete.

9. (*du, delegation end inquire received/response delegation End message, DR*): the delegating user *du* received the request to end the delegation, and responds with the delegation END message to *DR*.

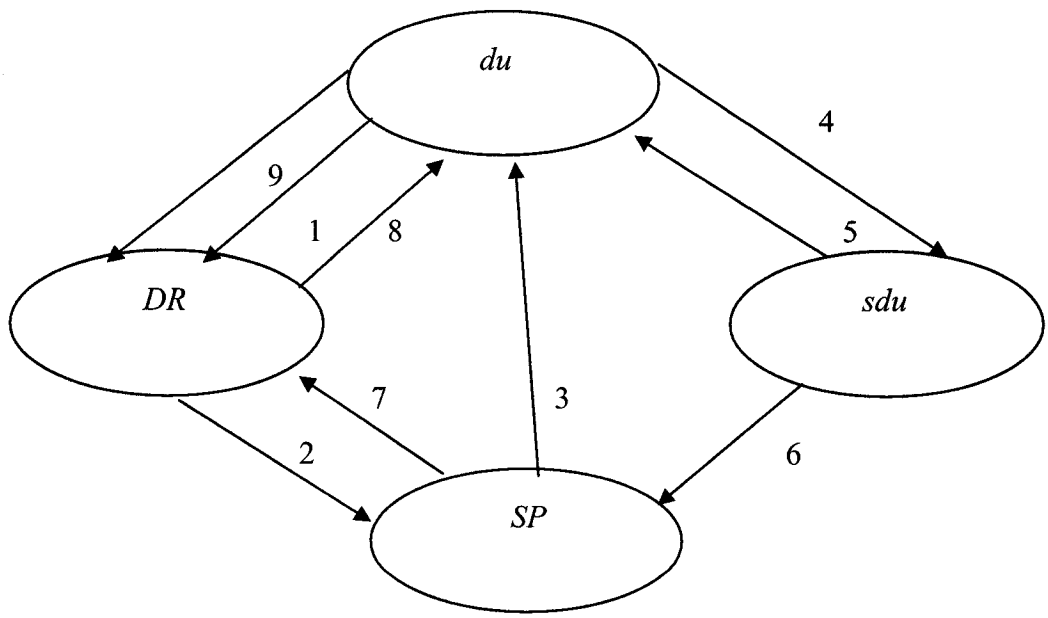

Figure 4-3. Delegation Process in FSM for Situation 2

4.6.2 A Proof of Delegation in FSM

i. *Complete.* Four states are involved in *situation 2* before delegation: a delegating user in a delegating role; a regular role, whose members will be selected and participate the delegation process; a delegated role, such that successful candidates are delegated into this role; a subset of permissions. Based on delegation requirements, a delegating user selects some members from a regular role, and delegates a subset of permissions associated with a delegated role to these members simultaneously. FSM in Figure 4-3 represents this process completely: transitions {1, 4, 5, 8, 9} support self-acted delegation; transitions {2, 3, 6} support partial delegation; transition 6 supports simultaneous delegation; transition 7 supports monotonic delegation; transition 5 supports bilateral delegation.

ii. *Minimal.* Four states are involved in *situation 2* when delegation occurs: a delegating user, a regular role, a subset of permissions, and a delegated role. Each pair of these states is distinguishable.

4.7 Delegation in *situation 3*

Before delegation, the candidates are members of different roles. Successful candidates must match the delegation requirements. We propose the following definition.

Definition 6

- N_1: the total number of delegated roles involved in the delegation process, $i \in N_1$.
- PC_1: all memberships of a regular role to be delegated, where $pc_i \in PC_1$.
- DAE_2: a set of attribute expressions referring to the delegation requirements for delegated candidates with different delegated roles.
- $DAE_2 = \{<r_1, expr_1, n_1>, <r_2, expr_2, n_2>, <r_3, expr_3, n_3> \ldots <r_i, expr_i, n_i>\}$, where $<r_i, expr_i, n_i>$ means the number of successful candidates of Role r_i who match the delegation requirements $expr_i$ is n_i.
- DW_1: a number that refers to the maximum delegation width, where $dw_1 = n_1 + n_2 + \ldots + n_i$.

- delegated_users DU_1: a set of users delegated to the delegated role, these users are members of different roles before delegation, $DU_1 \subset U$, $Users\_D(r) \subset DU_1$, $DU_1 \subseteq PC$.
- $CU = OU \cup DU_1$.
- $can\_delegate_2 \subseteq R \times PC_1 \times \{DW_1\} \times \{DD\} \times DAE_2 \times DT$.
- $SDR \subseteq UA \times UA$ is a one-to-many delegation relation. This delegation relation can be expressed by $((u, r), (DU_1, r')) \in SDR$.

The meaning of $(r, pc_i, dw_1, dd, dae_1, dt) \in can\_delegate_2$ is that a member of role r (or a role senior to r) can delegate role r (or a role junior to role r) to a set of other users who are members of different regular roles sdu_i and satisfy the prerequisite condition pc_i and dae_1 without exceeding the maximum delegation width dw_1, the maximum delegation depth dd, and the duration constraint dt. In this thesis, dd=1. Delegation expression $expr_i$ can be a set of attributes if applicable. Let's see one example. if (*PROF1, {RA1,RA2}, 3, 1,{<RA1,PHD,2>, <RA2,PHD,1>}, FALL2005*) $\in$ can_delegate$_2$, then Professor Martin can delegate *PDF1* to Lisa, Mike and Tina simultaneously during the fall term 2005; Lisa and Mike are PHD students, and members of *RA1*; Tina is a PHD student, and a member of *RA2* before delegation; so that (*Martin, PROF1, {Lisa, Mike, Tina}, PDF1*) $\in$ *SDR*. Lisa, Mike and Tina cannot delegate the same role to anyone else (single-step delegation). If $dw_1 = 1$, then URDM1 carries out a single delegation.

4.7.1 Delegation Process in FSM

A least three states are in the delegation process when *situation 3* is involved: *du, DR* and sdu_i. We address the following 12 state-transitions in Figure 4-4.

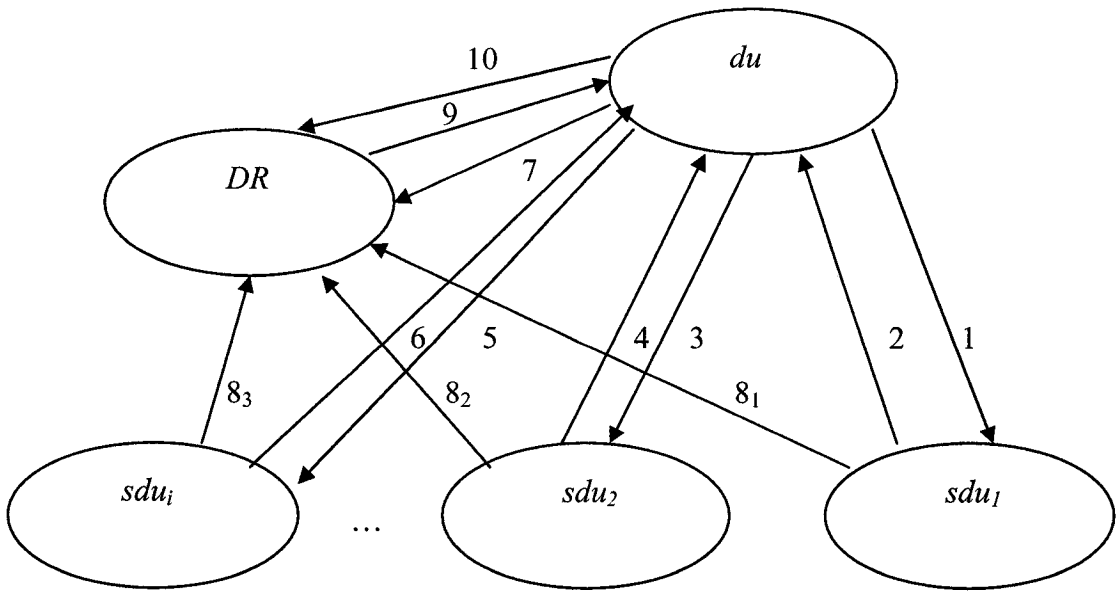

Figure 4-4. Delegation Process in FSM for Situation 3

1. (*du*, *delegation request/candidates searching*, sdu_1): a delegating user *du* sends out the delegation requests and will start to look for delegated candidates in sdu_1.

2. (sdu_1, *request received/searching accepted, du*): a set of delegated candidates in sdu_1 receive the delegation request and respond to *du* to accept the delegation.

3. (*du*, *delegation request/candidates searching*, sdu_2): the delegating user *du* sends out the delegation requests and will start to look for delegated candidates in sdu_2.

4. (sdu_2, *request received/searching accepted, du)*: a set of delegated candidates in sdu_2 receive the delegation request and respond to *du* to accept the delegation.

5. (*du*, *delegation request/candidates searching*, sdu_i): the delegating user *du* sends out the delegation requests and will start to look for delegated candidates in sdu_i.

6. (*sdu_i, request received/searching accepted, du*): a set of delegated candidates in *sdu_i* receive the delegation request and respond to *du* to accept the delegation process.

7. (*du, new members acceptance request/ready for acceptance, DR*): *du* sends the accept- new-members request to the delegated role *DR*, *DR* is ready to accept new members.

8_1. (*sdu_1, successful candidates selected/successful candidates added, DR*): a set of candidates in *sdu_1* have been selected and added into the delegated role *DR*.

8_2. (*sdu_2, successful candidates selected/successful candidates added, DR*): a set of candidates in *sdu_2* have been selected and added into the delegated role *DR*.

8_3. (*sdu_i, successful candidates selected/successful candidates added, DR*): a set of candidates in *sdu_i* have been selected and added into the delegated role *DR*.

9. (*DR, new members acceptance/delegation end inquire, du*): the delegated role *DR* sends the *new member acceptance* message and informs *du* that the delegation is complete.

10. (*du, delegation end inquire received/response delegation End message, DR*): the delegating user *du* receives the request to end the delegation, and responds with the delegation END message.

Transitions $\{8_1, 8_2, 8_3\}$ occur simultaneously when simultaneous delegation is running in URDM.

4.7.2 A Proof of Delegation in FSM

i. *Complete.* At least three states are involved in *situation 3* before delegation: a delegating user in a delegating role; a set of regular roles, whose members will be selected and participate the delegation process; a delegated role, such that successful candidates are delegated into this role. Based on delegation requirements, a delegating user selects some members from a set of regular roles, and these members are delegated into a delegated role simultaneously. FSM in Figure 4-4 represents this process completely: transitions {1, 2, 3, 4, 5, 6, 7, 9, 10} support self-acted delegation; transitions $\{8_1, 8_2, 8_3\}$ support simultaneous delegation; transitions {2, 4, 6} support bilateral delegation.

ii. *Minimal.* At least three states are involved in *situation 3* when delegation occurs: a delegating user in a delegating role, a set of regular roles, and a delegated role. Each pair of these states is distinguishable.

4.8 Delegation in *Situation 4*

URDM can handle the following simultaneous delegation as well: the delegating user can delegate subsets of permissions in *du* at a time to a set of delegated users, who are members of different sdu_i.

Definition 6 and Definition 7 are used to support simultaneous delegation in *situation 4*. When the delegation relation is required, only *SDR'* is involved in this delegation.

The meaning of $(r, pc_i, dw_1, dd, dae_1, dt) \in can\_delegate_2$ is that a member of role *r* (or a role senior to *r*) can delegate role *r* with a set of permissions (or a role with a set of permissions junior to role *r*) to a set of other users who are members of different regular roles sdu_i and satisfy the prerequisite condition pc_i and dae_1 without exceeding the maximum delegation width dw_1, the maximum delegation depth *dd*, and the duration constraint *dt*. In this thesis, *dd*=1. Delegation expression $expr_i$ can be a set of attributes if applicable. Let's see one example. if *(PROF1, {RA1,RA2}, 3, 1,{<RA1,PHD,2>,*

<*RA2,PHD,1*>}, *FALL2005*) ∈ can_delegate$_2$, then Professor Martin can delegate *PDF1* with a subset of permissions {*lab access, conference attendance guaranteed*} to Lisa, Mike and Tina simultaneously during the fall term 2005; Lisa and Mike are PHD students, and members of *RA1*; Tina is a PHD student, and a member of *RA2* before delegation; so that (*Martin, PROF1,* {*Lisa, Mike, Tina*}, *PDF1*<-{*lab access, conference attendance guaranteed*}) ∈ *SDR'*. Lisa, Mike and Tina cannot delegate the same role to anyone else (single-step delegation). If $dw_1 = 1$, then URDM1 carries out a single delegation. If Professor Martin delegates *PDF1* with all permissions associated with *PDF1*, this simultaneous delegation is identical to the delegation in *situation 3*.

4.8.1 Delegation Process in FSM

At least four states are involved in such a delegation process: *du, DR, SP* and sdu_i. This FSM includes the following 15 state-transitions (see Figure 4-5.).

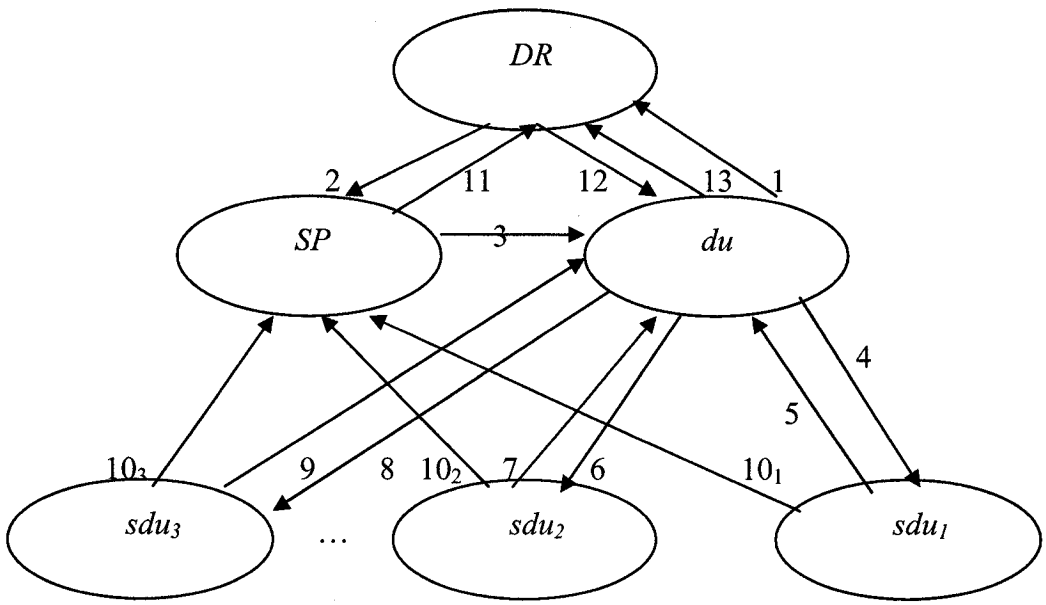

Figure 4-5. Delegation in FSM for Situation 4

1. (*du, new members conditional-acceptance request/conditional acceptance request received, DR*): *du* sends the conditional-accept-new-members request to the delegated role *DR; DR* knows the request.

2. (*DR, delegated permissions searched/delegated permission ready, SP*): According to *du* request, *DR* searches the permissions that will be delegated and collects them for delegation.

3. (*SP, delegated permission check request/delegated permission matched, du*): *SP* sends the delegated permission check request to *du;* the collected permissions by *DR* match the request of *du*.

4. (*du, delegation request/candidates searching,* sdu_1): the delegating user *du* sends out the delegation requests and will start to look for delegated candidates in sdu_1.

5. (sdu_1, *request received/searching accepted, du*): a set of delegated candidates in sdu_1 receive the delegation request and respond to *du* to accept the delegation.

6. (*du, delegation request/candidates searching,* sdu_2): the delegating user *du* sends out the delegation requests and will start to look for delegated candidates in sdu_2.

7. (sdu_2, *request received/searching accepted, du*): a set of delegated candidates in sdu_2 receive the delegation request and respond to *du* to accept the delegation.

8. (*du, delegation request/candidates searching,* sdu_i): the delegating user *du* sends out the delegation requests and will start to look for delegated candidates in sdu_i.

9. (sdu_i, *request received/searching accepted, du*): a set of delegated candidates in sdu_i receive the delegation request and respond to *du* to accept the delegation.

10_1. (*sdu_1, successful candidates selected/ permissions delegated, SP*): a set of candidates have been selected from *sdu_1* and the collected permissions are delegated.

10_2. (*sdu_2, successful candidates selected/ permissions delegated, SP*): a set of candidates have been selected from *sdu_2* and the collected permissions are delegated.

10_3. (*sdu_i, successful candidates selected/ permissions delegated, SP*): a set of candidates have been selected from *sdu_i* and the collected permissions are delegated.

11. (*SP, permissions that have been delegated informed/permissions arrangement, DR*): *DR* knows that permissions have been delegated and arranges the current permission use list after delegation.

12. (*DR, new members acceptance/delegation end inquire, du*): the delegated role *DR* sends the *new member with conditional acceptance* message and informs *du* that the delegation process needs to end.

13. (*du, delegation end inquire received/response delegation End message, DR*): the delegating user *du* received the request to end the delegation, and responds with the delegation END message. (This transition is necessary, because the delegating user controls the delegation process in URDM).

Transitions $\{10_1, 10_2\ 10_3\}$ occur simultaneously when simultaneous delegation is running in URDM.

4.8.2 A Proof of Delegation in FSM

i. *Complete.* At least four states are involved in *situation 4* before delegation: a delegating user in a delegating role; a set of regular roles, whose members will be

selected and participate the delegation process; a delegated role, such that successful candidates are delegated into this role; a subset of permissions. Based on delegation requirements, a delegating user selects some members from a set of regular roles, and delegates a subset of permissions associated with a delegated role to these members simultaneously. FSM in Figure 4-5 represents this process completely: transitions {1, 3, 4, 5, 6, 7, 8, 9, 12, 13} support self-acted delegation; transitions {2, 10_1, 10_2 10_3} support partial delegation; transitions {10_1, 10_2 10_3} support simultaneous delegation; transitions {10_1, 10_2 10_3, 11} support monotonic delegation; transitions {5, 7, 9} support bilateral delegation.

ii. *Minimal.* At least four states are involved in *situation 4* when delegation occurs: a delegating user in a delegating role, a set of regular roles, a subset of permissions, and a delegated role. Each pair of these states is distinguishable.

4.9 Summary

Analysis of situations before delegation is a very important step for simultaneous delegation in URDM. Before delegation, the delegated users may be members of the same regular role *sdu,* or members of different roles sdu_i. At the same time, the delegating user may plan to delegate only subsets of permissions that are associated with the delegated role.

In this chapter, URDM has been proposed to support simultaneous delegation based on the above situations. Because existing delegation models in the literature have not formalized the delegation process, we have used the Finite State Machine (FSM) technique to clarify this process.

Chapter 5 Revocation in URDM

5.1 Introduction

Revocation is the reverse process of delegation. It refers to the process by which a delegating user can take away the privileges or the delegated role that he/she delegated to another user who is a member of another role. Revocation is a necessary part of delegation management in URDM.

In this chapter, we will address the following topics related to revocation:

- Introduction to Revocation
- Types of Revocation
- Revocation Supported in URDM

5.2 Introduction to Revocation

So far we have addressed how a user in a delegating role can delegate his/her role or permissions to other users in other roles and how we can control this process using the relation *can_delegate*. However, in practice, revoking the delegated role or the delegated permissions is a necessary part of delegation management in RBAC. In the examples that we have analyzed in the previous chapters, Professor Martin is in charge of a research project at the PHD level, and he delegated the role *PDF* or permissions associated with *PDF* to some students in role RA_i during the Fall Semester 2005. When this project ends, Professor Martin has to take away the delegated role *PDF* (or those permissions) from the delegated users. Such a process is referred as a revocation process.

5.3 Types of Revocation

Revocation is the reverse process of delegation. According to different revocation purposes, the following types of revocation have been proposed:

i. Duration-based Revocation. This type of revocation means that the delegation is temporary and expires with time; the length of the delegation becomes critical to the effectiveness of delegation. Duration-based Revocation has two advantages: This is a simple self-triggering process that ensures the revocation of delegate membership automatically; and because of attaching a timeout stamp to the delegation process, we no longer have to worry about tracking the delegating user. However, this timeout stamp must be chosen carefully. Over-estimating the duration of delegation increases risk by allowing the delegated members to continue to execute the permissions or the roles assigned to the delegated members; and under-estimating the duration means that the delegated users may not have sufficient time to complete the tasks for which they were originally delegated the permissions.

ii. User-based Revocation. This type of revocation means that the revocation processes are managed by users instead of a timeout stamp. This user-based revocation can be categorized into the following detailed types.

- Cascading Revocation. This type refers to the indirect revocation of membership as a result of the revocation of the membership of some other related roles. With the regular role *sdu* as we defined it in the previous chapter, if the delegated member loses her membership in her regular role, then as a result she will lose her delegated membership in the delegated role *DR*. In the case of the delegating user in the delegating role, if this delegating role is revoked, then the membership that was delegated by that revoked role will also be revoked. This can be considered as a chain-based revocation.

- Non-Cascading Revocation. This type of revocation only revokes a delegated user assignment.

- Grant-Independent Revocation. This type of revocation allows any original member in a delegating role to revoke the membership of any delegated member in the delegated role. This gives the power to the original members to protect the role from the temporary delegate members, which can have some advantages and some disadvantages. An advantage is that in the case where the delegate member behaves badly, any original member can revoke him immediately which will minimize the damage before even the time out. A disadvantage on the other hand is that it raises the possibility of conflicts between the original members. This can occur if someone else other than the granting original member revokes the delegate membership.

- Grant-Dependent Revocation. This type of revocation means that only the delegating member is allowed to revoke the role or permissions that he or she delegated. This revocation makes the process of revocation more

controllable: it eliminates the possibility of conflict between the original members.

- Strong Revocation. This revocation of a user from a role requires that the user be removed not only from the delegated role but also from the regular role before delegation.

- Weak Revocation. This revocation only removes the user from the delegated role and leaves other roles intact.

5.4 Revocation Supported in URDM

In large-scale environments, URDM supports Grant-Dependent Revocation (GD). RAE is defined in URDM which is a set of the attribute expressions for a revocation purpose. We also use the definitions in Definition 3 to represent a revoke delegation relation called *can_revoke*.

Definition 7

- RAE : a set of attribute expressions for revocation requirements.
- *can_ revoke* $\subseteq$ R $\times$ CU $\times$ RAE $\times$ DT.

The expression (*r, cu, rae, dt*) $\in$ *can_ revoke* means that within all current users in the delegated role after delegation (either an original user *ou* or a delegated user *du*), if a delegated user *du* satisfies the revocation attributes, then only the delegating user can revoke this user from current users in the delegated role. For example, if (*PROF1, PDF1,* {*<PHD, delegated>*}, *FALL2005*) $\in$ *can_revoke*, then Professor Martin can revoke Mike from *PDF1*. Mike is a delegated user in *PDF1,* and he is a PHD student during the fall term 2005.

5.4.1 Revocation in FSM

We assume that there are two states involved in the revocation process: *du* and *DR*. Five transitions (see Figure 5-1.) are addressed as follows:

1. (*du, revocation request/candidate searching, DR*): the delegating user *du* sends the revocation request and will start to search for revocation candidates in *DR*.

2. (*DR, revocation request received/ready for revocation, du*): the delegated role *DR* receives the revocation request and is ready for revocation.

3. (*du, successful revocation candidate selected/the candidate revoked, DR*): the delegating user selects the candidate for revocation from *DR* and revokes this candidate from *DR*.

4. (*DR, revocation end inquire/ready for revocation end, du*): the delegated role *DR* sends the revocation END message to *du* and is ready for revocation end.

5. (*du, revocation end inquire received/response the End, DR*): the delegating user *du* receives the inquiry from *DR*, and sends back the END message to *DR*.

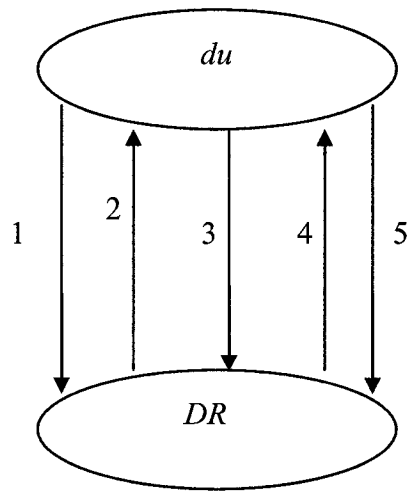

Figure 5-1. Revocation in FSM

5.4.2 A Proof of Revocation in FSM

i. *Complete.* Two states are involved in revocation: a delegating user in a delegating role; a delegated role. Based on revocation requirements, a delegating user selects a delegated member from this delegated role, and removes this person from the delegated role. It refers to a GD revocation. FSM in Figure 5-1 represents this process completely.

ii. *Minimal.* Two states are involved when revocation occurs: a delegating user and a delegated role. This pair of states is distinguishable.

5.5 Summary

Revocation is the process by which a delegation that was accepted is removed or retracted. It is a very important process in delegation management.

In this chapter, we have introduced the concept of revocation. Two types of revocation have been represented: Duration-based Delegation and User-based Delegation. User-based Delegation includes Cascading/Non-Cascading Revocation, Grant-Independent /Grant-Dependent Revocation, and Strong/Weak Revocation. URDM is designed to support Grant-Dependent Revocation. A revocation relation called can_ *revoke* is also shown. The revocation process is analyzed in FSM as well.

Chapter 6 Design and Implementation

6.1 Introduction

URDM supports simultaneous delegation, role hierarchy and single step delegation. In chapter 4, we addressed URDM within four situations: a delegating user can delegate a set of users into a delegated role at the same time, these users are members of the same role before delegation; a delegating user can delegate a set of users into a delegated role with specific permissions simultaneously, these users are members of the same role before delegation; a delegating user can delegate a set of users into a delegated role simultaneously, these users are members of different roles before delegation; and a delegating user can delegate a set of users into a delegated role with specific permissions simultaneously, these users are members of different roles before delegation.

In this chapter, we will design and implement the core functionality of URDM within the first situation. This implementation is a web application. The following topics will be discussed:

- Introduction to the Implementation Environment.
- Application Design.
- A Sample of Implementation Results.

6.2 Introduction to the Implementation Environment

We implement a part of URDM by using Dreamweaver 8. The implementation techniques include ASP with VB Script, ADO, Microsoft Access 2003 and SQL. Internet Information Services 5.1 (IIS 5.1) must be installed and activated before implementation. We use a local host to demonstrate our model.

6.2.1 Dreamweaver 8

Dreamweaver 8 is the industry-leading web development tool, enabling users to efficiently design, develop and maintain standards-based websites and applications. With Dreamweaver 8, web developers go from start to finish, creating and maintaining basic websites to advanced applications that support best practices and the latest technologies. Because our application is a web system, we choose Dreamweaver 8 as our design platform.

Computer System requirements for Dreamweaver 8 include [Yank 2006]:

Windows

- 800 MHz Intel Pentium III processor (or equivalent) and later
- Windows 2000, Windows XP
- 256 MB RAM (1 GB recommended to run more than one Studio 8 product simultaneously)
- 1024 x 768, 16-bit display (32-bit recommended)
- 650 MB available disk space

Macintosh

- 600 MHz PowerPC G3 and later
- Mac OS X 10.3, 10.4
- 256 MB RAM (1 GB recommended to run more than one Studio 8 product simultaneously)

- 1024 x 768, thousands of colors display (millions of colors recommended)
- 300 MB available disk space

6.2.2 Internet Information Services 5.1

Internet Information Services (IIS) 5.1 [Internet Information Services 2006] is a powerful Web server that provides a highly reliable, manageable, and scalable web application infrastructure for all versions of Windows Server 2003. IIS 5.1 helps organizations increase web site and application availability while lowering system administration costs. IIS 5.1 supports the Microsoft Dynamic Systems Initiative (DSI) with automated health monitoring, process isolation, and improved management capabilities. The benefits of deploying IIS 5.1 include less planned and unplanned system downtime, increased web site and application availability, lower system administration costs, server consolidation, and a significant increase in web infrastructure security. In order to run Dreamweaver 8 sucessfully, IIS 5.1 must be installed first.

6.2.3 ASP

Activate Server Pages (ASP) is the server-based technology from Microsoft, designed to create dynamic and interactive HTML pages for those developers' World Wide Web site or corporate intranet. Instead of using the browser to locate the web page, ASP uses another machine –the web server–before returning the results to the user as HTML.

ASP [Birmingham 1998] was officially announced to the world by Microsoft on July 16, 1996, codenamed "Denali". It gained much wider recognition when it was bundled with version 3 of Microsoft's Internet Information Server Suite in March 1997. ASP was considerably enhanced with the release of IIS 5.1, and now offers an enriched model for managing communications between a browser and a web server. Dreamweaver 8 supports the ASP technique for web design.

6.2.4 VBScript

Scripting languages provide the web developers with a more accessible gateway to programming. Client-side scripting for web use was developed to provide a dynamic alternative to static HTML. When a browser finds a scripting instruction embedded in HTML code, the browser will translate that script into pure HTML. This permits a developer to create more interactive web pages, which are far more functional than pure HTML pages. Scripting languages form the basis of ASP. ASP scripts are executed by the web server, rather than the browser. VBScript [VBScript Tutorial 2006] is Microsoft's scripting language, and is based on their Visual Basic programming language. We design our web application by using ASP with VBScript.

6.2.5 ADO

Microsoft ActiveX Data Objects (ADO) [Microsoft ActiveX Data Objects 2006] is a set of Component Object Model objects for accessing data sources. It provides a layer between programming languages and OLE DB (a means of accessing data stores, whether they be databases or otherwise, in a uniform manner), which allows a developer to write programs which access data, without knowing how the database is implemented. ADO consists of several top-level objects:

- Connection (represents the connection to the database)
- Recordset (represents a set of database records)
- Command (represents a SQL command)
- Record (represents a set of data, typically from a source other than a database)
- Stream (represents a stream of data, as from a text file or web page)
- Error (stores errors)
- Field (represents a database field)
- Parameter (represents a SQL parameter)
- Property (stores information about objects)

The ADO components are usually used in conjunction with a high-level language such as VBScript in an ASP environment.

6.2.6 Microsoft Access 2003

MS Access 2003 provides a powerful set of tools that are sophisticated enough for professional developers to create or use powerful database solutions that make organizing, accessing, and sharing information easier than ever [Access 2003 Production Description 2006]. MS Access 2003 is used to create a database for our web application.

6.2.7 SQL

Structured Query Language (SQL) [James 2006] is a standardized language designed to access and manipulate data stored in relational databases and to work with the databases themselves. An SQL statement will most commonly contain coded instructions to perform one or more of the following operations:

- Extracting specific data from the database
- Inserting new data into the database
- Modifying existing data
- Deleting data

We extract the records from database by using SQL and display these records on the web pages.

6.3 Application Design

Before we start to design this web application, some assumptions are necessary to address.

6.3.1 Assumptions

The example that has been defined in chapter 3 will continue to be used in this chapter. This web application is a University Delegation Management System (UDMS). In order to secure the delegation process, only the authorized delegating users (professors) can

login into UDMS. There are four roles in UDMS: a delegating role, namely Professors; a regular role before delegation, namely ResearchAssistants; a delegated role, namely PDF; and a pre-selected-candidates role before delegation, namely SR. We assume these four roles as four corresponding tables in a database called delegationBase.

6.3.2 Design

We will represent our design phase in this section.

i. Delegation:

a) Create a database. Open MS Access 2003 and create a database called delegationBase.

b) Create four tables, namely Professors, ResearchAssistants, PDF, and SR. We can see the attributes of these tables in Table 6-1.

Professors				
ProfID	ProfName	Academic Unit	Prof_passwd	UserName
1	Martin	software engineering	2345	martin

Table 6-1 (a). Professors Table

ResearchAssistants			
ResearchID	RAName	RAStatus	RASemester
R10	martin	master	fall
R11	kate	phd	fall
R12	peter	master	winter
R13	lina	phd	winter
R14	tina	phd	fall
R90	david	phd	winter

Table 6-1 (b). Research Assistant Table

PDF			
ID	Name	Status	Semester
P500	Jack	phd	fall

Table 6-1(c). PDF Table

SR			
ResearchID	RAName	RAStatus	RASemester

Table 6-1 (d). SR Table

c) Design a login webpage called Index.asp. Open Dreamweaver 8 and create Index.asp. The common way to access a database from inside an ASP page is to:

- Create an ADO connection to a database
- Open the database connection
- Create an ADO recordset
- Open the recordset
- Extract the data you need from the recordset
- Close the recordset
- Close the connection

Let's see the above process step by step.

- Create an ADO connection to a database. The easiest way to connect to a database is to use a DSN-less connection. A DSN-less connection can be used against any Microsoft Access database on your web site.

```
<%
set conn=Server.CreateObject("ADODB.Connection")
conn.Provider="Microsoft.Jet.OLEDB.4.0"
conn.Open "c:/Inetpub/wwwroot/delegationBase.mdb"
%>
```

- Open the database connection, create an ADO recordset and open the recordset.

```
<%
set rs=Server.CreateObject("ADODB.recordset")
rs.Open "Professors", conn
%>
```

- Extract the data you need from the recordset. Before login, we assume the authorized delegating user is Professor Martin, his user name is martin, and password is 2345 and this data has been defined in Professors Table. A brief algorithm for the login phase is below:

```
{ENTER username & password
Click Login Button
If username = username in table Professors & password = Prof_passwd in table
Professors  Then
Response.redirect "delegationzone.asp"
Else
Response.Write "index.asp?errMsg=User name and password are incorrect"
}
```

- Close the recordset and close the connection.

```
<% rs.close
   conn.close%>
```

d) Design delegationzone.asp

- The similar ways to access a table called Research Assistants from inside an ASP.
- SELECT all records of Research Assistants Table, display these records.
- SELECT all records of PDF Table, display these records.

- Create two delegation constraints text fields, one is called Delegation Attribute, in which are defined the potential delegated candidates' status (our example is PHD); the other is called Delegation During: fall semester or winter semester.
- SELECT those records which match the inputs from the two above text fields.
- Store these selected records into a temporary table called SR. This SR must be empty before the delegation process.
- In this chapter, we demonstrate simultaneous delegation in URDM by delegating two candidates into the delegated role at the same time. Hence, we create two series of text fields with information about the two pre-selected candidates. Only those users in SR can be delegated. One single delegation is supported as well during this step.

e) Design delegationresult.asp

- After the above inputs, click the delegation button.
- Open PDF.
- INSERT those pre-selected simultaneous delegated users into PDF.
- All records of PDF are displayed, including those delegated users.
- Remove all records in SR.
- Close SR and PDF.
- Close connection.

ii. Revocation:

a) Continue in delegationresult.asp: Because UDMS supports GD revocation, only the delegating user can revoke the delegated users, we continue our revocation process by the end of the delegation for authorization purpose.

- Setup a button, to determine whether the revocation process is required or not. If this button is clicked, the webpage will be redirected to a new page called revocation.asp.

b) Design revocation.asp

- Display all members of PDF after delegation.
- Create two revocation attribute fields to determine pre-selected revocation candidates. One is called Attribute 1, in which are defined the potential revocation candidates' status (our example is PHD); the other is called Attribute 2 in which are defined the potential revocation candidates' semester (our example is winter).
- Display all pre-selected candidates: Because a prefix of a delegated member's ID in PDF is "R", and a prefix of original member's ID in PDF is "P", only delegated members in PDF are displayed.
- Create one series of text fields with information about the pre-selected candidate. Only those pre-selected delegated members can be accepted and revoked.
- Create a button for running revocation. This button is to redirect to the last page called revocationresult. Asp

c) Design revocationresult.asp

- Display all members in PDF after revocation.

6.4 A Sample of Design, Implementation and Result Interfaces

We show a sample design interface in Figure 6-1.

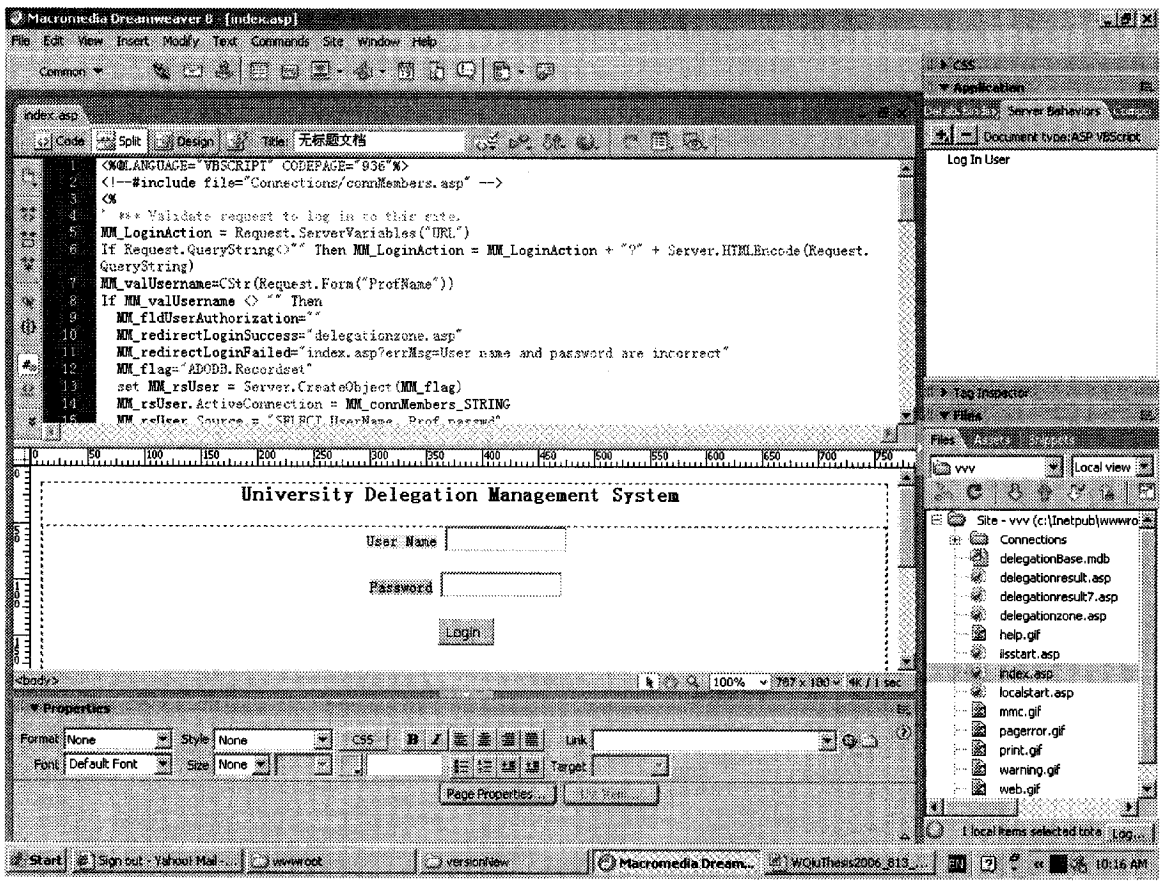

Figure 6-1. A Sample Design Interface

We demonstrate the login phase in Figure 6-2, the failed login in Figure 6-3, the delegation candidates of Research Assistants and members of PDF before pre-selection in Figure 6-4, two attributes for delegation in Figure 6-5, the candidates of Research Assistants after pre-selection in Figure 6-6, the invalid input and result of delegation in Figure 6-7, and the valid inputs and result of simultaneous delegation in Figure 6-8.

We also show the revocation interface in Figure 6-9, two attributes for revocation in Figure 6-10, display the pre-selection in Figure 6-11, the invalid input and result of revocation in Figure 6-12, and the valid input and result of revocation in Figure 6-13.

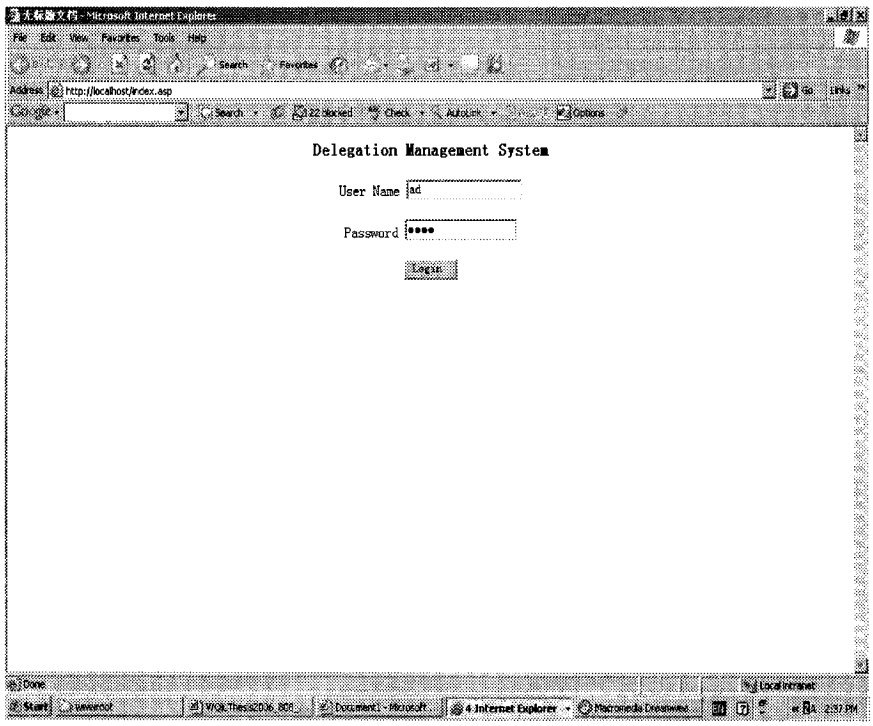

Figure 6-2. A Sample Login Interface

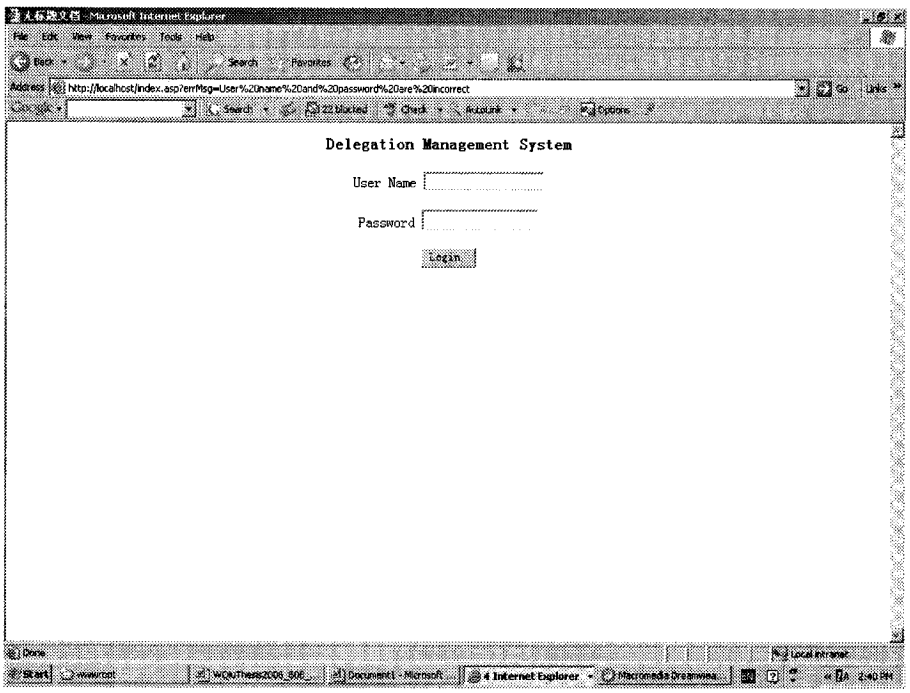

Figure 6-3. Failed Login

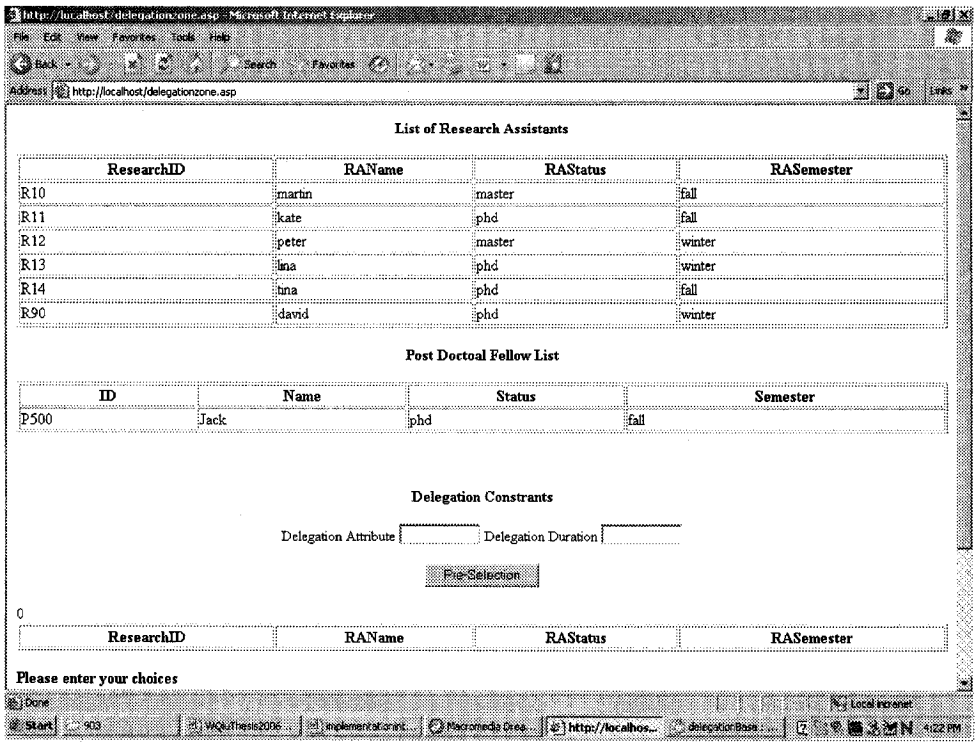

Figure 6-4. Members of ResearchAssistants and PDF before Delegation

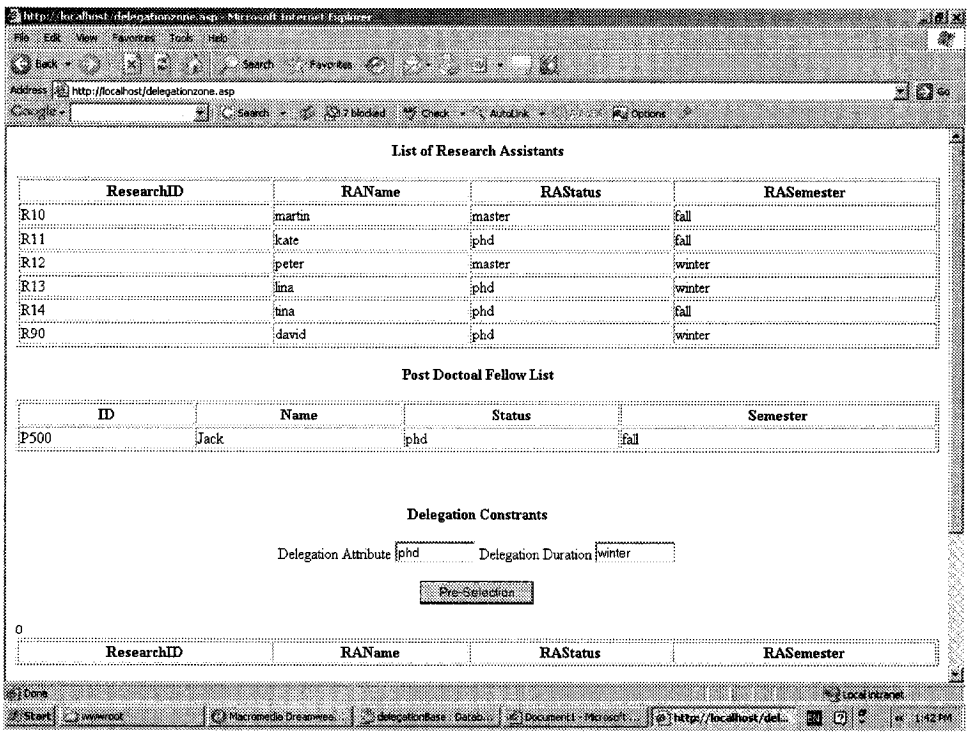

Figure 6-5. Two Attributes for Delegation

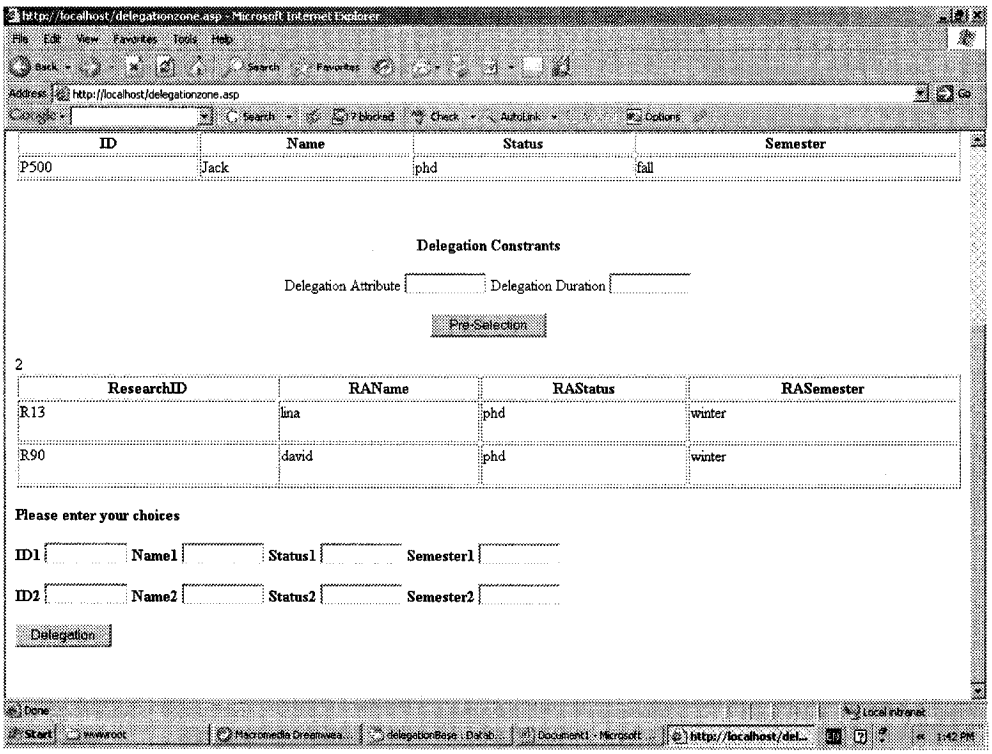

Figure 6-6. Candidates after Pre-Selection

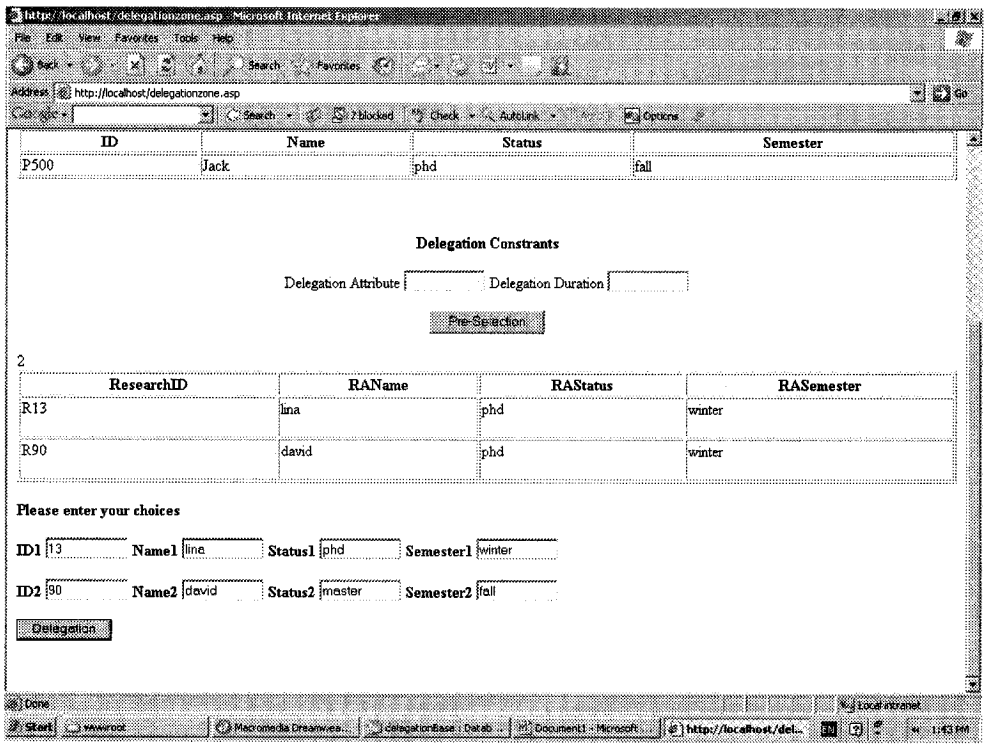

Figure 6-7(a). Invalid Input for delegation

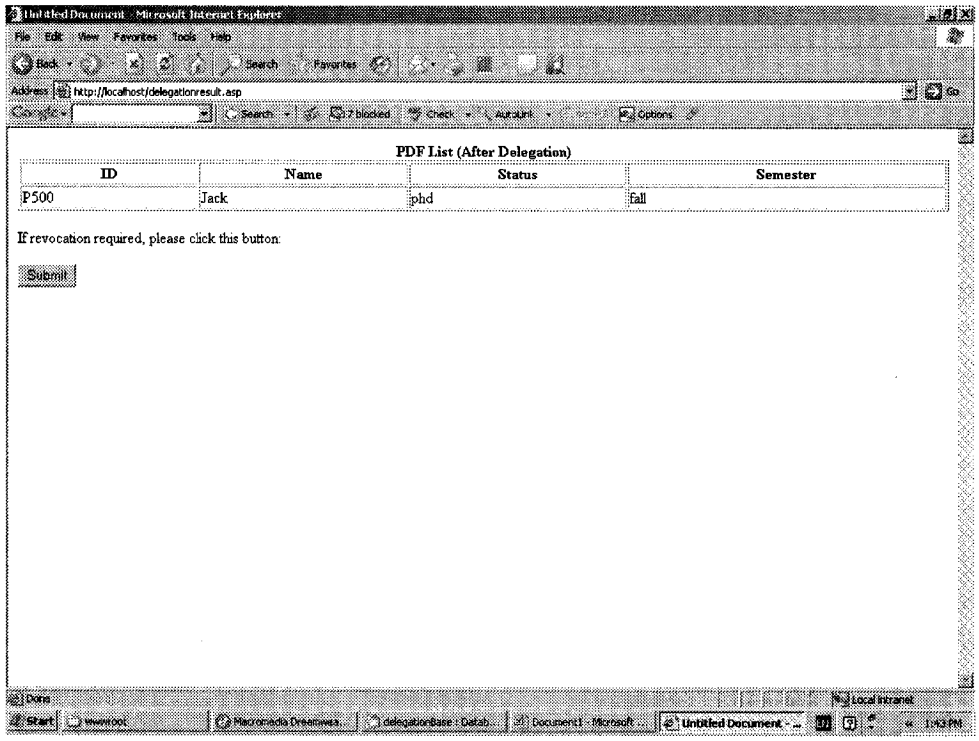

Figure 6-7(b). Result of Delegation after Invalid Input

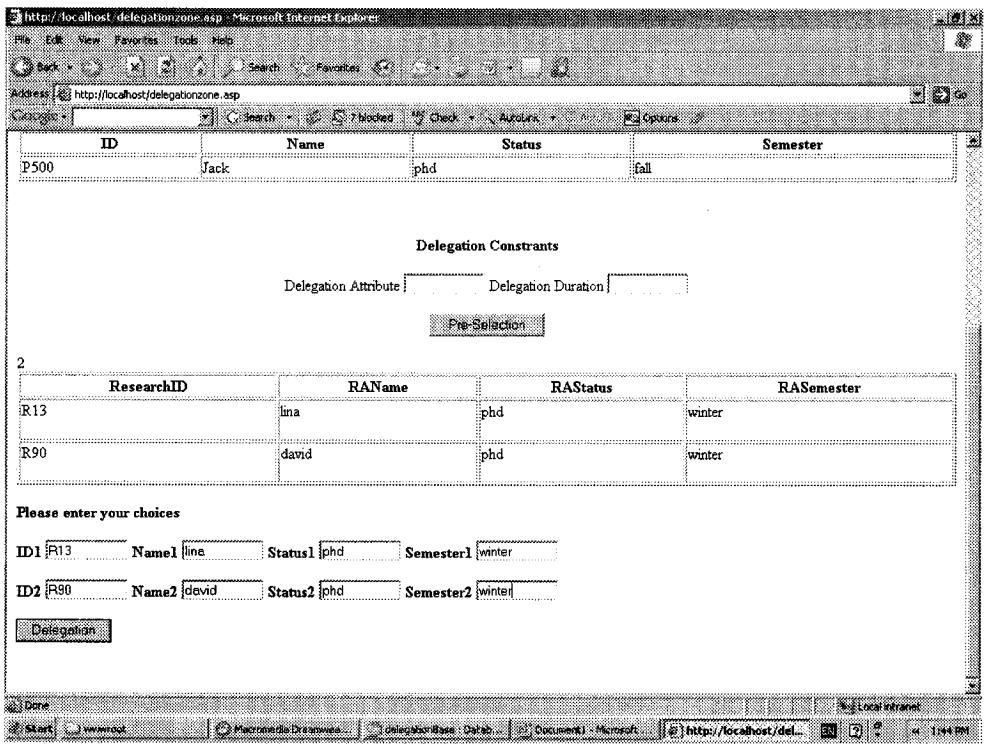

Figure 6-8(a). Valid Inputs before Delegation

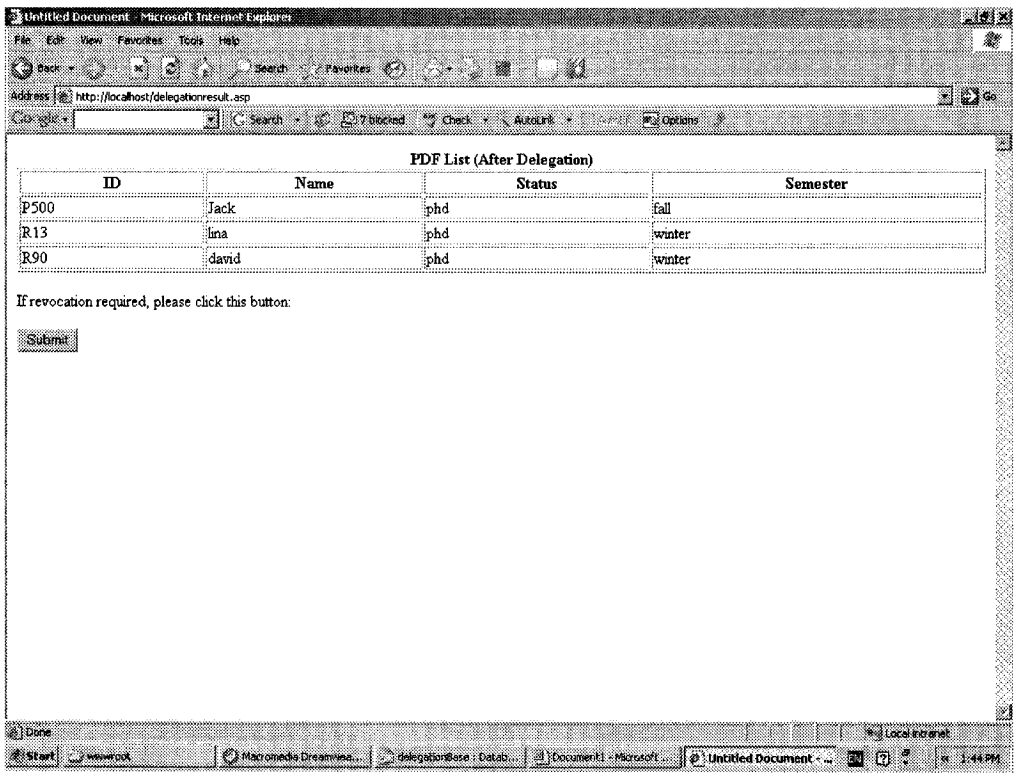

Figure 6-8 (b). Result of Valid Delegation

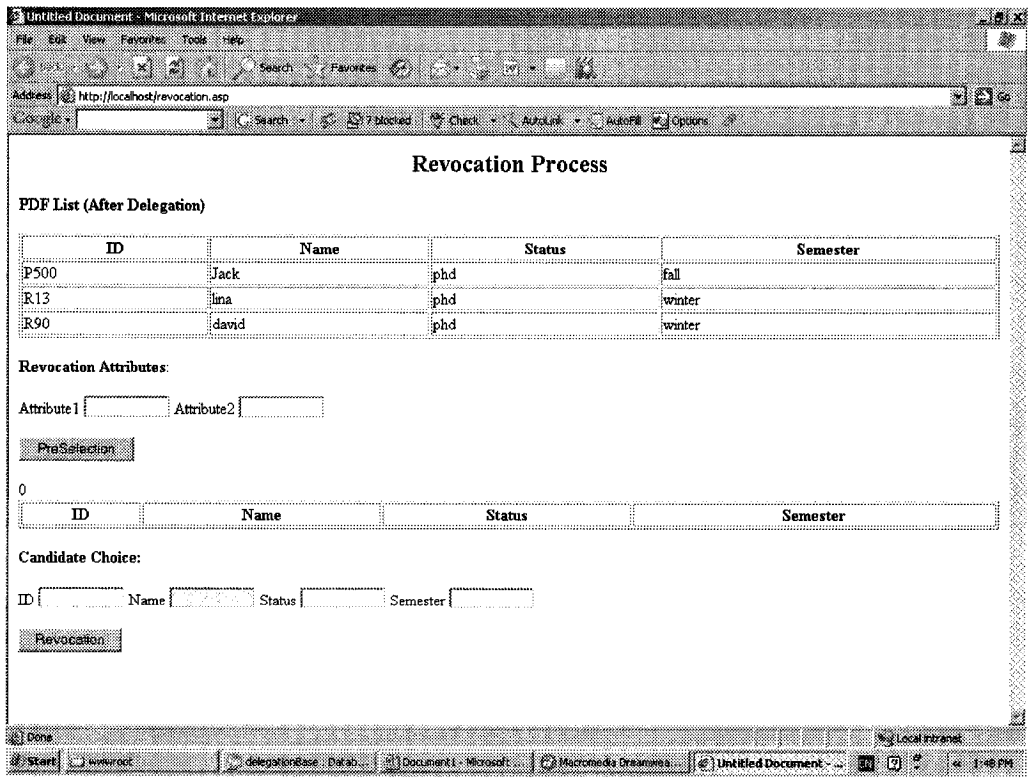

Figure 6-9. A Revocation Interface

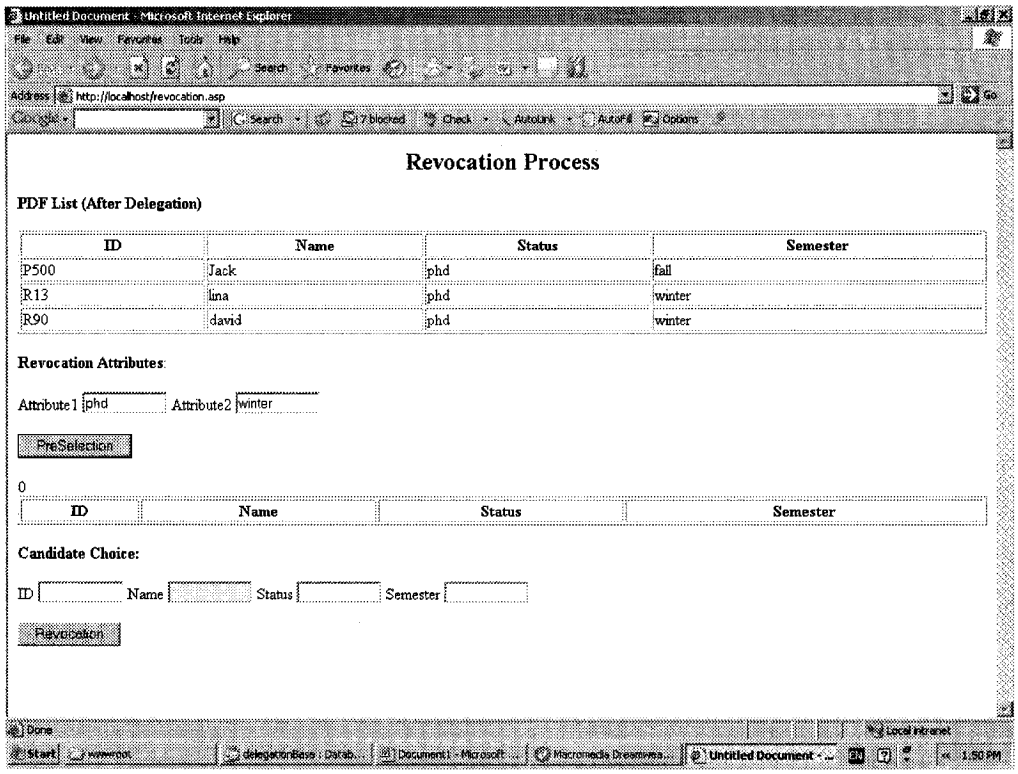

Figure 6-10. Two Attributes for Revocation

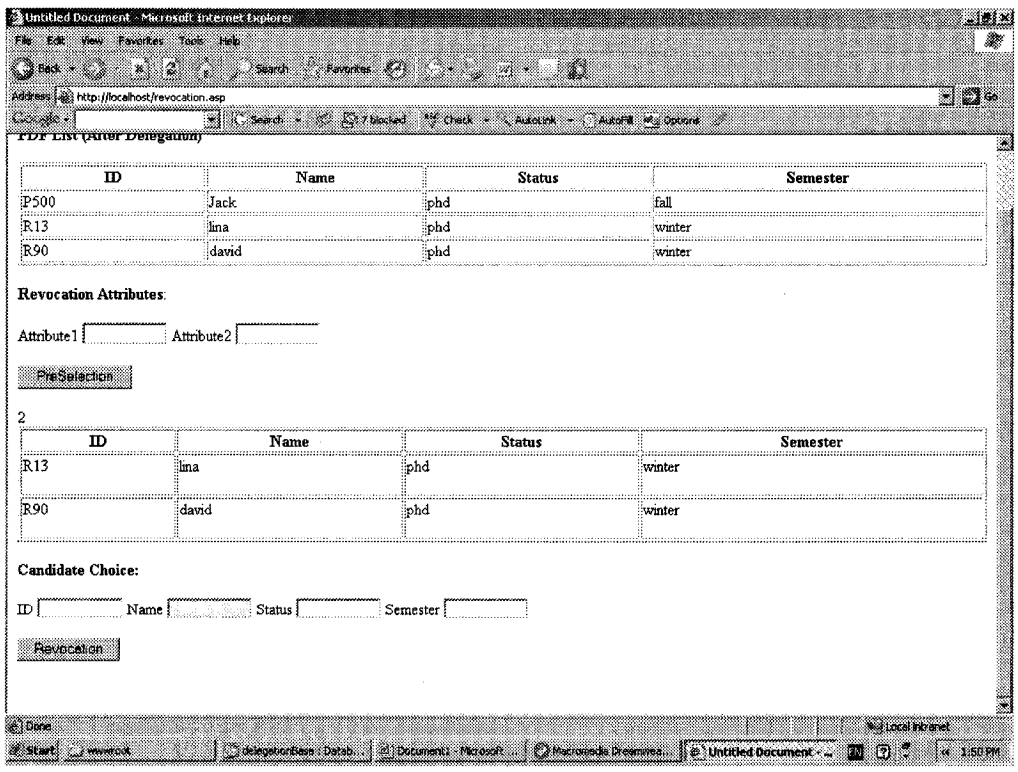

Figure 6-11. Display the Pre-Selection Result

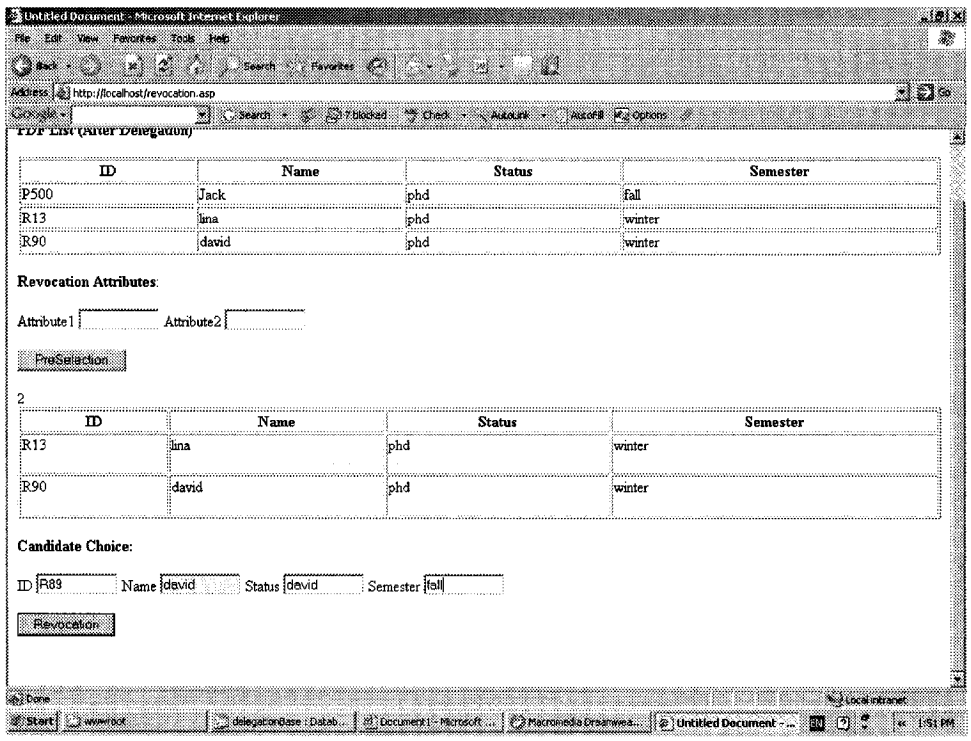

Figure 6-12 (a). Invalid Input for Revocation

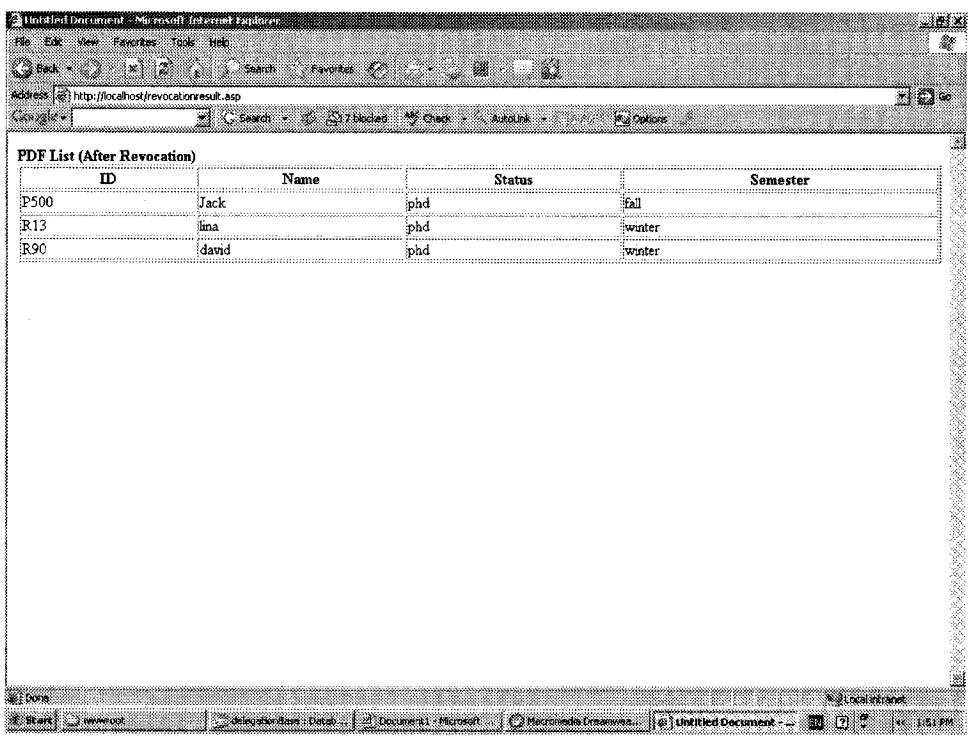

Figure 6-12(b). Result of Invalid Revocation

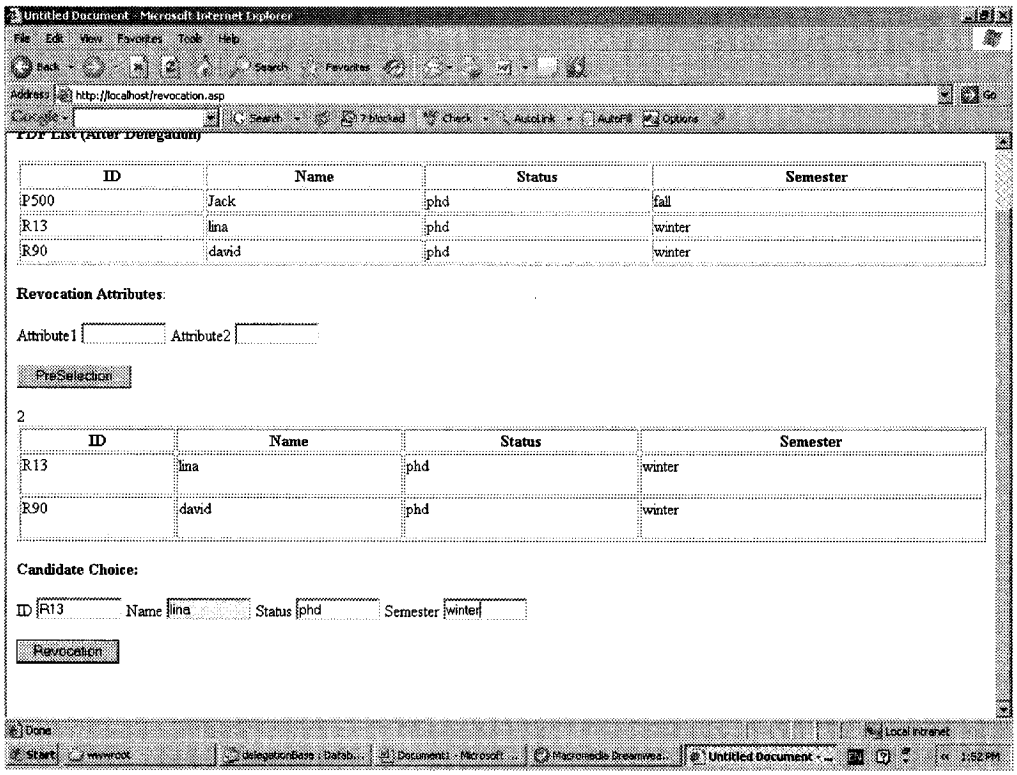

Figure 6-13 (a). Valid Input for Revocation

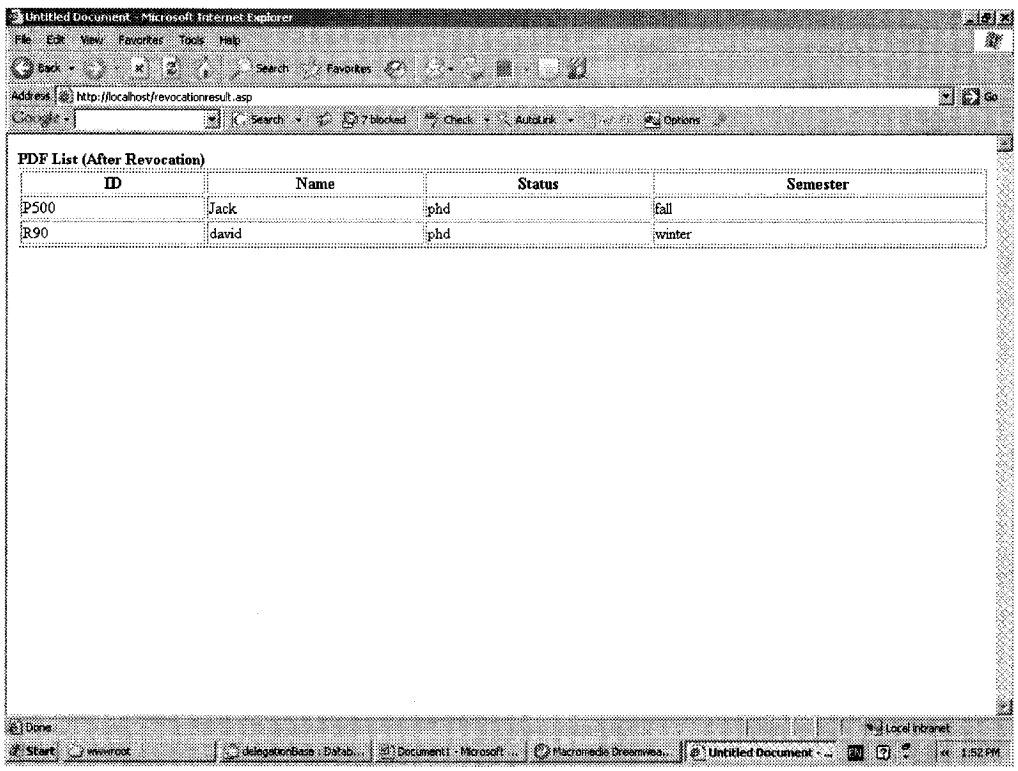

Figure 6-13 (b). Result of Valid Revocation

6.5 Summary

URDM is designed to support simultaneous delegation, role hierarchy and single step delegation within four situations. In this chapter, we have designed and demonstrated the implementation of URDM in the first situation. We used a software tool called Dreamweaver 8 to design a web application. Before the implementation, Internet Information Services 5.1 must be installed. The web application is called University Delegation Management System (UDMS). The programming techniques include ADO, MS Access 2003, SQL, ASP and VBScript. Samples of the implementation results have been presented.

Chapter 7 Conclusion and Future Work

7.1 Introduction

A delegation process is to allow a junior role, or a role which is not included in the hierarchy, to be temporarily granted senior roles' permissions in RBAC. URDM has been proposed in this thesis to support simultaneous delegation, role hierarchy and single step delegation.

In this chapter, we will conclude this thesis and suggest directions for future work.

7.2 Conclusion

In information technology, authorization is concerned with the ways in which users can access resources in the computer system. Access control is arguably the most fundamental and most pervasive security mechanism in use today. Access control shows up in virtually all systems and imposes great architectural and administrative challenges at all levels of enterprise computing.

RBAC, one of a variety of access control technologies, was created to make commercial security policies easier to manage. Like multi-level security, RBAC is conceptually simple: access to computer system objects is based on a user's role in an organization. However, a junior role, or a role which is not included in the hierarchy, cannot perform the tasks of a senior role. In order to allow this when required, various delegation models have been proposed to allow junior roles to be temporarily granted senior roles' permissions, such as PBDM, RDM2000 and ABDM. However, RDM2000 does not support simultaneous delegations. We have proposed a new delegation model called User-to-Role Delegation Model (URDM), which is an extension of RDM2000. URDM supports simultaneous delegations, role hierarchy, and single-step delegation in four situations: the delegating user cannot delegate arbitrary subsets of permissions but can only delegate the entire role at a time to a set of delegated users, who are members of the same role before delegation; the delegating user can delegate subsets of permissions associated with the delegated role at a time to a set of delegated users, who are members of the same role before delegation; the delegating user can only delegate an entire role at a time to a set of delegated users, who are members of different roles before delegation; the delegating user can delegate subsets of permissions associated with the delegated role at a time to a set of delegated users, who are members of different roles before delegation. URDM supports Grant-Dependent Revocation as well. We have implemented a web application called University Delegation Management System (UDMS) to illustrate the first situation.

Zhang, et al., [Zhang et al. 2001], proposed RDM2000 to support hierarchical roles and sub-delegation. However, one important challenge remained in RDM2000: a delegating user may wish to delegate a set of permissions to all members of another role at the same time. This problem was identified as simultaneous delegation in [Barka and Sandhu 2002]. As a new delegation model, URDM has achieved both expressivity and performance improvements to solve this simultaneous delegation issue in RBAC.

7.3 Future Work

We have proposed URDM to support simultaneous delegation, role hierarchy and single step delegation. Because single step delegation is a special case of sub-delegation, it will be useful to consider how sub-delegation can be achieved in URDM. Cascading revocation in URDM is another interesting direction for future research in this area. We have also demonstrated URDM by developing a web application called UDMS. Future work may extend our framework to support information sharing in other applications, such as healthcare systems, government and commercial organizations. The application of delegation in RBAC to healthcare to enforce legal access requirements is common. Emergencies demand that doctors who may not normally be permitted to a patient's records be allowed to see these records to save their life. How to solve such a situation in URDM will be an interesting direction for further work. On the other hand, URDM has been proposed to support simultaneous delegation in a single system; how can trust be built up between an external party and the information system? We hope to extend the architecture of URDM for cross-organization authorization in our future work. In the era of data transfer mobility, how to handle new and complex problems of mobile and distributed access to resources and information sharing will be another interesting direction to explore.

References

Access Control Methodologies, Jones and Bartlett web site.2006. http://computerscience.jbpub.com/chapple/chapple02.pdf#search=%22rule-based%20access%20control%22, accessed February 2006.

Access 2003 Production Description. 2006. MSDN Academic Alliance Software Center website, http://msdn06.e-academy.com/elms/Storefront/ViewProductWriteup.aspx?campus=uoo_ite&p=506&pwt=6, accessed June 2006.

Alotaiby, Fahad T., Chen, J. X. 2004. *A Model for Team-based Access Control (TMAC 2004,* International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 1, 2004.

Answers web site.2006. http://www.answers.com/topic/delegation, accessed April 2006.

Barka, Ezedin. 2002. *Thesis: Framework for Role-Based Delegation Model.* George Mason University, Fairfax, Virginia, USA. 2002.

Barka, Ezedin and Sandhu, Ravi. 2000a. *Framework for Role-Based Delegation Model.* Proceedings of 16th Annual Computer Security Application Conference, Sheraton New Orleans, December 11-15, 2000.

Barka, Ezedin and Sandhu, Ravi. 2002b. *A Role-Based Delegation Model and Some Extensions*, 2000.

Barka, Ezedin, Aly, Alaa, Kudaimi, Wadhah, Hayawi, Kadem, *Implementation of Role-Based Delegation Model/Flat Roles (RBDM0)*, the Sixth CIT - 90h Annual U.A.E. University Research Conference.

Birmingham. 1998. *ASP 2.0 Programmer's Reference*, Wrox Press, England, 1998.

Chen, Fang and Sandhu, S. R. 1996. *Constraints for Role-Based Access Control*, ACM RBAC Workshop. MD,1996.

Chou, Shih-Chien. 2004. *L^nRBAC: A Multiple-Levelled Role-Based Access Control Model for Protecting Privacy in Object-Oriented Systems*, Journal of Object Technology, vol. 3, no. 3, pp. 91-120, March-April 2004.

Cohen, Eve, Thomas, Roshan K., Winsborough, William and Shands, Deborah. 2002. *Models for coalition-based access control (CBAC)*, Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies, Monterey, California, USA, 2002.

Computing Dictionary web site. 2006. http://computing-dictionary.thefreedictionary.com/Mandatory+access+control, accessed March 2006.

Coyne, E. J. 1996. *Role engineering.* In Proceedings of the ACM Workshop on Role-Based Access Control, 1996.

Ferraiolo, F. David, Kuhn, D. Richard and Chandramouli, Ramaswamy, 2003. Role-Based Access Control. Library of Congress Cataloguing in Publication Data. ARTRCH HOUSE, Inc., 2003.

Gasser, Morrie and McDermott, Ellen.1990. *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.

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

Internet Information Services 5.1. 2006. Microsoft Windows XP web site, http://www.microsoft.com/windowsxp/evaluation/features/iis.mspx, accessed June 2006.

James Hoffman. 2006. *Introduction to Structured Query Language*, http://www.geocities.com/SiliconValley/Vista/2207/sql1.html, accessed June 2006.

Leidigh, Christopher. 2005. *Fundamental Principles of Network Security*, American Power Conversion, 2005.

Microsoft ActiveX Data Objects. 2006. w3 schools web site, http://www.w3schools.com/ado/default.asp, accessed June 2006.

NIST. 2006. National Institute of Standards and Technology web site, http://www.nist.gov/dads/HTML/finiteStateMachine.html, accessed June 2006.

Password, Wikipedia web site. 2006. http://en.wikipedia.org/wiki/Password, accessed March 2006.

R. K. Thomas and R. S. Sandhu. 1997. *Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management.* Proceedings of the IFIP WG11.3 Workshop on Database Security, Lake Tahoe, California, August 11-13, 1997.

Role Definition web site. 2006. http://www.agentlab.de/role_view.html, accessed March 2006.

Sandhu, R., Bhamidipati, V., and Munawer, O. 1999. *The ARBAC97 Model for Role-based Administration of Roles.* ACM Transactions on Information and System Security 2, Pages 105-135, February 1st, 1999.

Sandhu, Ravi. 1996a. *Rationale for the RBAC96 Family of Access Control Models*, ACM RBAC Workshop, MD, USA, 1996.

Sandhu, Ravi. 1996b. *Authentication, Access Control, and Audit*, ACM Computing Surveys, Vol. 28, No. 1, March 1996.

Sandhu, Ravi, Coyne, Edward J. and Youman, C. E.1996c. *Role-Based Access Control Models*. IEEE Computer, 29(2):38–47, February 1996.

Smartcard web site. 2006. http://www.kfupm.edu.sa/smartcard/whatis.htm, accessed April 2006.

Spence, Bill. 2003. *Biometrics' Role in Physical Access Control*, Loss Prevention & Security Journal, April 2003.

Stockley, Derek web site. 2006. http://derekstockley.com.au/newsletters-05/036-effective-delegation.html, accessed March 2006.

Thomas, R.K. and Sandhu, R.S. 1994. *Conceptual Foundations for A Model of Task-based Authorizations.* Proceedings of the IEEE Computer Security Foundations Workshop, New Hampshire, IEEE Press, 1994.

Ural, Hasan. 2004. Course notes for *CSI5174: Valid Methods in Distributed Systems*, School of Information and Technology Engineering, University of Ottawa, Fall, 2004.

VBScript Tutorial. 2006. w3 schools web site, http://www.w3schools.com/vbscript/default.asp, accessed June 2006.

Yank, Kevin. 2006. *Review - Dreamweaver 8 (Macromedia)*, sitepoint web site, http://www.sitepoint.com/article/dreamweaver-8-review, accessed June 2006.

Ye, Chunxiao, Fu, Yunging and Wu, Zhongfu. 2004. *An Attribute-Based Delegation Model.* InfoSecu04, Shanghai, China, November14-16, 2004.

Yialelis, N., Sloman, M. S. 1996. *A Security Framework Supporting Domain Based Access Control in Distributed Systems*, ISOC Symposium on Network and Distributed System Security (SNDSS96), February, 1996.

Zhang, L., Ahn, G. and Chu, B. 2001. *A Rule-Based Framework for Role-Based Delegation.* Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT), pages 153-162. Chantilly, VA, May 3-4, 2001.

Zhang, Xinwen, Oh, Sejong and Sandhu, Ravi. 2003. *PBDM: A Flexible Delegation Model in RBAC.* Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT), Como, Italy, June 2-3, 2003.