A framework for role-based group delegation in distributed environments

Hua Wang Jiuyong Li Ron Addie Stijn Dekeyser Richard Watson

Department of Maths & Computing, University of Southern Queensland
Toowoomba QLD 4350 Australia
Email: (wang, jiuyong, addie, dekeyser, rwatson)@usq.edu.au

Abstract

Role-based delegation model (*RBDM*) based on the role-based access control (*RBAC*) has proven to be a flexible and useful access control model for information sharing in a distributed collaborative environment. In today's highly dynamic distributed systems, a user often needs to delegate a role to all members of a group at the same time. It presents the challenge of how to build a role-based group delegation framework within *RBAC* in distributed environment.

This paper aims to build a group delegation framework within *RBAC*. The framework includes a role-based group delegation granting model, group delegation revocation model, granting authorization and revocation authorization. We analyze various revocations and the impact of revocations on role hierarchies. The implementation with *XML* based tools demonstrates the framework and authorization methods. Finally, comparisons with other related work are indicated.

1 Introduction

The National Institute of Standards and Technology developed the role-based access control (*RBAC*) prototype (Feinstein H. L. 1995) and published a formal model (Ferraiolo D. F. and Kuhn D. R. 1992). *RBAC* has been widely used in database system management and distributed environments since it enables managing and enforcing security in large-scale and enterprise-wide systems. *RBAC* involves individual users being associated with roles as well as roles being associated with permissions (Each permission is a pair of objects and operations). As such, a role is used to associate users and permissions. A user in this model is a human being. A role is a job function or job title within the organization associated with authority and responsibility.

Permission is an approval of a particular operation to be performed on one or more objects. As shown in Figure 1, the relationships between users and roles, and between roles and permissions are many-to-many (i.e. a permission can be associated with one or more roles, and a role can be associated with one or more permissions). The security policy of the organization determines role membership and the allocation of each role's capabilities.

The *RBAC* model supports the specification of several aspects.

a. User/role associations – the constraints specifying user authorizations to perform roles;

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

c. Duty separation constraints – these are role/role associations indicating conflict of interest:

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

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

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

The number of roles and users in a large enterprise system can be hundreds or thousands. Managing these roles and users, and their interrelationships is a vital challenge that is often highly decentralized and delegated to a small team of project groups. User-role assignment is a particularly critical administrative activity because assigning people to tasks is a normal managerial function and assigning users to roles is a natural part of assigning users to tasks. Furthermore, the activities in distributed environment are decentralized and delegated to users rather than system administrators.

Delegation is an important aspect of *RBAC* and often regarded as one of the principal motivation behind *RBAC*. Although the importance of delegation in *RBAC* has been recognozed for a long time, it has not received much attention. Zhang et al (Zhang L., Ahn G., and Chu B. 2001, Zhang L., Ahn G., and Chu B. 2002) recently proposed a rule-based framework for role-based delegation including the *RDM2000* model. We use the concept of role-based delegation borrowed from their work. The central contributions of this article are to describe how we can build a framework of role-based group delegation within *RBAC* in a distributed environment and to implement the delegation framework with *XML* based tools and languages.

The remainder of this paper is organized as follows: Section 2 presents the related work associated with the delegation models and *RBAC*. As the results of this section, we find that both group-based delegation within *RBAC* and its implementation with *XML* has not been analysed and presented in the literature. Section 3 proposes a delegation framework which includes the structures of role-based delegation and role-based group delegation models. Section 4 provides delegation authorizations. Granting authorization with pre-requisite conditions and revocation authorization are discussed in this section. Definitions of $Can\_delegate$, Can_revoke, *role range* are introduced. Section 5 describes the implementation of

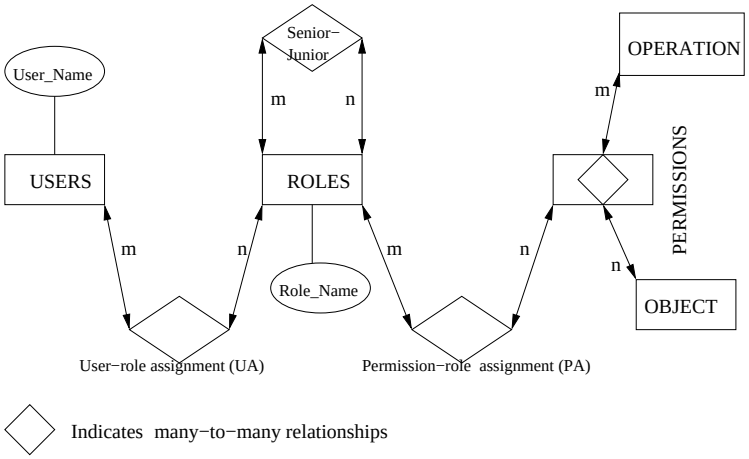

Figure 1: RBAC relationship.

the role-based group delegation using *XML* technology and Section 6 compares the work in this paper and related previous work. Finally, the conclusion of the paper is in Section 6.

2 Related work

Delegation is an important feature in many collaboration applications. For example, imagine that the Immigration Department is developing partnerships between immigration agencies and people in local areas to address possible problems. Immigration officers are able to prevent illegal stays and crime if they efficiently collaborate with the people. The problem-oriented immigrating system (*POIS*) is proposed to improve the service as a part of the Immigration Department's ongoing community efforts including identifying potential problems and resolving them before they become significant. With efficient delegation, officers respond quickly to urgent messages and increase the time spent confronting problems.

In *POIS*, officers might be involved in many concurrent activities such as conducting initial investigations, analysing and confronting crimes, preparing immigration reports, and assessing projects. In order to achieve this, users may have one or more roles such as lead officer, participant officer, or reporter. In this example, Tony, a director, needs to coordinate analysing and confronting crimes and assessing projects. Collaboration is necessary for information sharing with members from these two projects. To collaborate closely and make two projects more successful, Tony would like to delegate certain responsibilities to Christine and her staff. The prerequisite conditions are to secure these processes and to monitor the progress of the delegation. Furthermore, Christine may need to delegate the delegated role to her staff as necessary or to delegate a role to all members of a group at the same time. Without delegation skill, security officers have to do excessive work because of their involvement in every single collaborative activity. The major requirements of role-based delegation in this example are:

1. Group-based delegation means that a delegating user may need to delegate a role to all members of a group at the same time.
2. Multistep delegation occurs when a delegation can be further delegated. Single-step delegation means that the delegated role cannot be further delegated.
3. Revocation schemes are an important feature of collaboration systems. They take away the delegated permissions. There are different revoking schemes; among them are strong and weak revocations, cascading and noncascading revocations, as well as grant-dependent and grant-independent revocations (Wang H., Cao J. and Zhang Y. 2003).
4. Constraints are an important factor in *RBAC* for laying out higher-level organizational policies (Wang H., Cao J. and Zhang Y. 2001). They define whether or not the delegation or revocation process is valid.
5. Partial delegation means only subsets of the permissions are delegated while total delegation means all permissions are delegated. Partial delegation is an important feature because it allows users only to delegate required permissions. The well-known least privilege security principle can be implemented through partial delegation.

Although the concept of delegation is not new in authorizations (Aura T. 1999, Barka E. and Sandhu R. 2000a, Wang H., Zhang Y., Cao J. and Varadharajan V. 2003, Wang H., Sun L., Zhang Y., and Cao J. 2005), role-based delegation received attention only recently (Barka E. and Sandhu R. 2000a, Barka E. and Sandhu R. 2000b, Zhang L., Ahn G., and Chu B. 2001, Zhang L., Ahn G., and Chu B. 2002). Aura (Aura T. 1999) introduced key-oriented discretionary access control systems that are based on delegation of access rights with public-key certificates. The systems emphasized decentralization of authority and operations but their approach is a form of discretionary access control. Hence, they can neither express mandatory policies like the Bell-LaPadula model (Bell D.E., La Padula L.J. 1976), nor is it possible to verify that someone does not have a certificate. Furthermore, some important policies such as separation of duty policies cannot be expressed with only certificates. They need some additional mechanism to maintain the previously granted rights and the histories must be updated in real time when new certificates are issued. Delegation is also applied in decentralized trust management (Blaze M. Feigenbaum J., Ioannidis J. and Keromytis A. 1999, Li N. and Grosof B. N. 2000). Blaze et al (Blaze M. Feigenbaum J., Ioannidis J. and Keromytis A. 1999) identified the trust management problem as a distinct and important component of security in network services and Li et al (Li N. and Grosof B. N. 2000) made a logic-based knowledge representation for authorization with tractable trust-management in large-scale, open, distributed

systems. Delegation was used to address the trust management problem including formulating security policies and security credentials, determining whether particular sets of credentials satisfy the relevant policies, and deferring trust to third parties. Other researchers have investigated machine to machine and human to machine delegations (Wang H., Cao J. and Zhang Y. 2001, Abadi M., Burrows M., Lampson B., and Plotkin G. 1993). For example, Wang et al (Wang H., Cao J. and Zhang Y. 2001) proposed a secure, scalable anonymity payment protocol for Internet purchases through an agent which provided a higher anonymous certificate and improved the security of consumers. The agent certified re-encrypted data after verifying the validity of the content from consumers. The agent is a human to machine delegation which can provide new certificates. However, many important role-based concepts, for example, role hierarchies, constraints, revocation were not mentioned.

Zhang et al (Zhang L., Ahn G., and Chu B. 2001, Zhang L., Ahn G., and Chu B. 2002) proposed a rule-based framework for role-based delegation including the *RDM2000* model. The *RDM2000* model is based on the *RBDM0* model which is a simple delegation model supporting only flat roles and single step delegation. Furthermore, as a delegation model, it does not support group-based delegation.

This paper focuses exclusively on a role-based delegation model which supports group-based delegation and its implementation with *XML* technology. We extend our previous work and propose a delegation framework including delegation granting and revocation models, group-based delegation. To provide sufficient functions with the framework, this paper analyses how changes to original role assignment impact upon delegation results. This kind of role-based group delegation and its implementation with *XML* have not been studied before.

3 Role-based group delegation framework

In this section we propose a role-based group delegation framework called *RBGDF* which supports role hierarchy and group delegation by introducing the delegation relation.

3.1 Role-based delegation model

A session is an important concept within *RBAC* which means a mapping between a user and possibly many roles. For example, a user may establish a session by activating some subset of assigned roles. A session is always associated with a single user and each user may establish zero or more sessions. There may be hierarchies within roles. Senior roles are shown at the top of the hierarchies. Senior roles inherit permissions from junior roles. Let $x > y$ denote x is senior to y with obvious extension to $x \geq y$. Role hierarchies provide a powerful and convenient means to enforce the principle of least privilege since only required permissions to perform a task are assigned to the role.

Although the concept of a user can be extended to include intelligent autonomous agents, machines, even networks, we limit a user to a human being in our model for simplicity.

Figure 2 shows the role hierarchy structure of *RBAC* in *POIS*. The following Table 1 expresses an example of user-role assignment in *POIS*.

There are two sets of users associated with role r:

Original users are those users who are assigned to the role r;

RoleName	UserName
DIR	Tony
HO1	Christine
HO2	Mike
Co1	Richard
Re1	John
CS	Ahn

Table 1: User-Role relationship.

Delegated users are those users who are delegated to the role r.

The same user can be an original user of one role and a delegated user of another role. Also it is possible for a user to be both an original user and a delegated user of the same role. For example, if Christine delegates her role *HO1* to Richard, then Richard is both an original user (explicitly) and a delegated user (implicitly) of role *Co1* because the role *HO1* is senior to the role *Co1*. The original user assignment (UAO) is a many-to-many user assignment relation between original users and roles. The delegated user assignment (UAD) is also a many-to-many user assignment relation between delegated users and roles.

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

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

1. $UAO \subseteq U \times R$ is a many-to-many original user to role assignment relation.
2. $UAD \subseteq U \times R$ is a many-to-many delegated user to role assignment relation.
3. $UA = UAO \cup UAD$.
4. Users: $R \Rightarrow 2^U$is a function mapping of roles to sets of users.
   $Users(r) = \{u|(u,r) \in UA\}$ where UA is user-role assignment.
5. $Users(r) = Users\_O(r) \cup Users\_D(r)$
   where
   $Users\_O(r) = \{u|\exists r' \geq r, (u,r') \in UAO\}$
   $Users\_D(r) = \{u|\exists r' \geq r, (u,r') \in UAD\}$

   $Users(r)$ includes all users who are members of role r. The users may be original and delegated users. The original users $Users\_O(r)$ are not only the member of role r but also the member of a senior role of r. The members in $Users\_D(r)$ are similar to that in $Users\_O(r)$.

With these components, we analyse group delegation in the remaining part of this section.

3.2 Role-based group Delegation

The scope of our model is to address user-to-user delegation supporting role hierarchies and group delegations. We consider only the regular role delegation in this paper, even though it is possible and desirable to delegate an administrative role.

A delegation relation (*DELR*) exists in the role-based delegation model which includes three elements: original user assignments *UAO*, delegated user assignment *UAD*, and constraints. The motivation behind this relation is to address the relationships among different components involved in a delegation. In a user-to-user delegation, there are five components: a delegating user, a delegating role, a delegated user, a delegated role, and associated constraints.

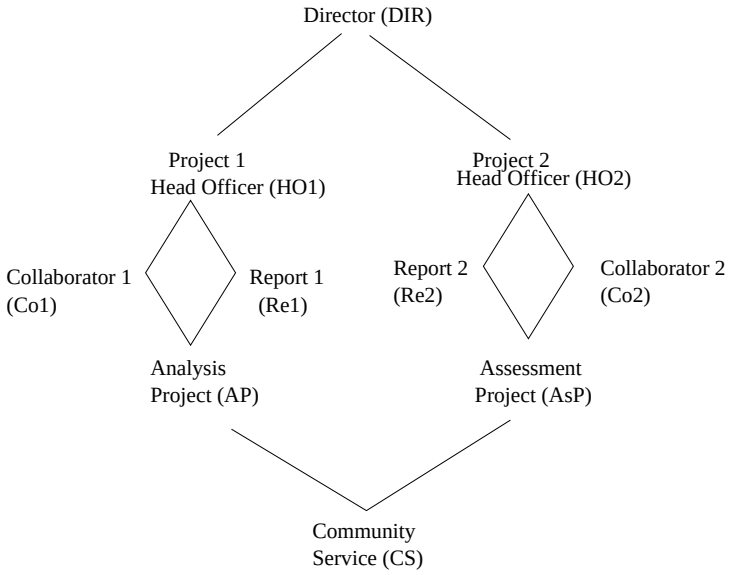

Figure 2: Role hierarchy in *POIS*.

For example, *((Tony, DIR), (Christine, DIR), Friday)* means Tony acting in role *DIR* delegates role *DIR* to Christine on Friday. We assume each delegation is associated with zero or more constraints. The delegation relation supports partial delegation in a role hierarchies: a user who is authorized to delegate a role r can also delegate a role r' that is junior to r. For example, *((Tony, DIR), (Ahn, Re1), Friday)* means Tony acting in role *DIR* delegates a junior role *Re1* to Ahn on Friday. A delegation relation is one-to-many relationship on user assignments. It consists of original user delegation (*ORID*) and delegated user delegation (*DELD*). Figure 3 illustrates components and their relations in a role-based delegation model.

From the above discussions, the following components are formalized:

1. $DELR \subseteq UA \times UA \times Cons$ is one-to-many delegation relation. A delegation relation can be represented by $((u, r), (u', r'), Cons) \in DELR$, which means the delegating user u with role r delegated role r' to user u' when the constraint $Cons$ is satisfied.
2. $ORID \subseteq UAO \times UAD \times Cons$ is an original user delegation relation.
3. $DELD \subseteq UAD \times UAD \times Cons$ is a delegated user delegation relation.
4. $DELR = ORID \cup DELD$

The last equation shows that delegation relations consist of original and delegated user delegation relations.

Now we analyse group delegation. In this paper we only discuss user-group delegations which consist of original user-group and delegated user-group delegations. The new relation of group delegation is defined as delegation group relation (*DELGR*) which includes: original user assignments *UAO*, delegated user assignments *UAD*, delegated group assignments *GAD*, and *constraints*. In a user-group delegation, there are five components: a delegating user (or a delegated user), a delegating role, a delegated group, a delegated role, and associated constraints. For example, *((Tony, DIR), (Project 1, DIR), 1:00pm–3:00pm Monday)* means Tony acting in role *DIR* delegates role *DIR* to all people involved in Project 1 during 1:00pm–3:00pm on Monday. A group delegation relation is one-to-many relationship on user assignments. It consists of original user group delegation (*ORIGD*) and delegated user group delegation (*DELGD*). Figure 4 illustrates components and their relations in role-based delegation model. Hence we have the following elements and functions in group delegation:

1. G is a set of users. GA is a set of group-role assignments.
2. $DELGR \subseteq UA \times GA \times Cons$ is one-to-many delegation relation. A delegation relation can be represented by $((u, r), (G, r), Cons) \in DELR$, which means the delegating user u with role r delegated role r to group G if the constraint $Cons$ is satisfied.
3. $ORIGD \subseteq UAO \times GAD \times Cons$ is a relation of an original user and a group with constraints.
4. $DELGD \subseteq UAD \times GAD \times Cons$ is a relation of a delegated user and a group with constraints.
5. $DELGR = ORIGD \cup DELGD$.

Based on the results of the structure with role-based group delegation, we discuss group delegation authorizations in the next section.

4 Delegation Authorization

We develop delegating and revocation models in this section. The notion of a *prerequisite condition*, *Can_delegate* and *Can_revoke* are key parts in group delegation process.

4.1 Authorization models

The delegation authorization goal imposes restrictions on which role can be delegated to whom. We partially adopt the notion of prerequisite condition from (Wang H., Cao J. and Zhang Y. 2003) to introduce delegation authorization in the delegation framework.

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

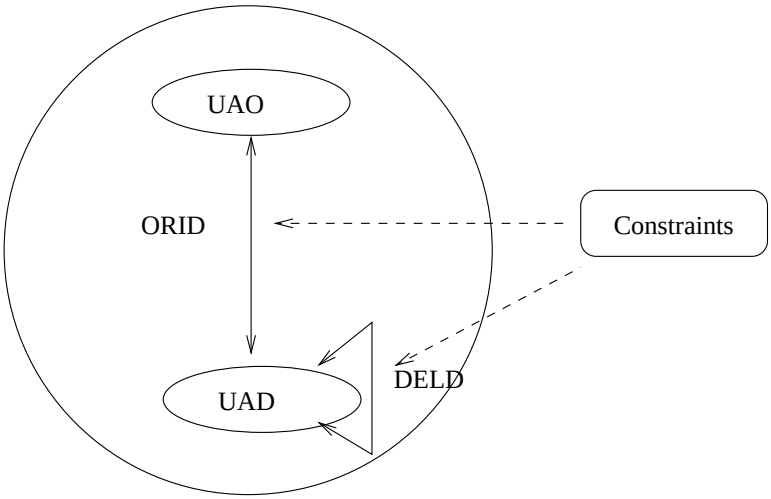

Figure 3: Role-based delegation model.

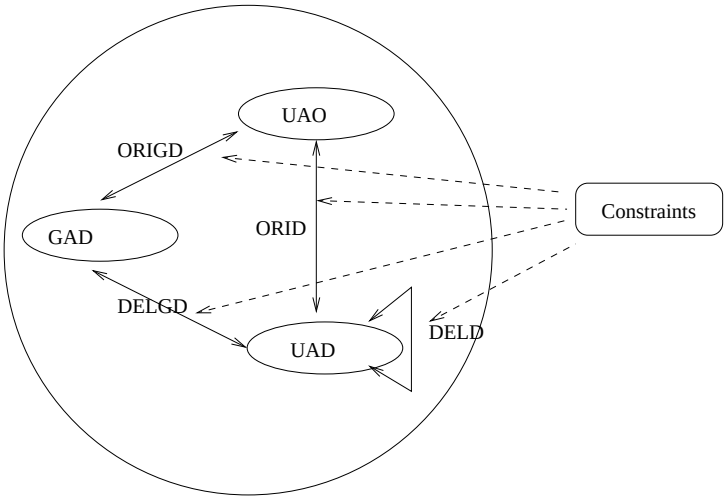

Figure 4: Role-based group delegation model.

We say a group satisfies a prerequisite condition if all users in the group satisfy the prerequisite condition.

For a given set of roles R let CR denote all possible prerequisite conditions that can be formed using the roles in R, for example, $CR = r_1 \wedge r_2 \vee \bar{r}_3$. In some cases, we may need to define whether or not a user can delegate a role to a group and for how many times, or up to the maximum delegation depth. Not every user can delegate a role to a user. The following relation provides what roles a user can delegate with prerequisite conditions.

Definition 1 *Can_delegate* is a relation of $R \times CR \times N$ where R, CR, N are sets of roles, prerequisite conditions, and maximum delegation depth, respectively. $\diamond$

The meaning of $(r, cr, n) \in Can\_delegate$ is that a user who is a member of role r (or a role senior to r) can delegate role r (or a role junior to r) to any group whose current entitlements in roles satisfy the prerequisite condition CR without exceeding the maximum delegation depth n. To identify a role range within the role hierarchy, the following closed and open interval notation is used.

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

User-group delegation is authorized by *Can_delegate*. Table 2 shows the *Can_delegate* relations with the prerequisite conditions in the *POIS* example. The meaning of *Can_delegate* $(DIR, [CS, HO1], 1)$ is that a member of role *DIR* can delegate role *DIR* and all roles in *POIS* (since all roles are junior to *DIR*) to a group whose current membership satisfies the prerequisite condition *[CS, HO1]* with one-step delegation. The second tuple authorizes that a user of role *HO1* can assign role *HO1*, and *Co1*, *Re1*, *AP* and *CS* to a group in which users are members of either role *AP*, or *Co1*, or *Re1* (since *AP*, *Co1* and *Re1* are in the range of *[AP, HO1))*.

RoleName (R)	Prereq.Condition (CR)	N
DIR	[CS, HO1]	1
HO1	[AP, HO1)	2
AP	CS	1
CS	ϕ	2

Table 2: Can delegate relations in POIS.

There are related subtleties that arise in *RBGDF* concerning the interaction between delegating and revocation of user-group delegation membership and the role hierarchy.

Definition 2 A user-group delegation revocation is a relation $Can\_revoke \subseteq R \times 2^R$, where R is the

set of roles. ◇

The meaning of $Can\_revoke(x, Y)$ is that a member of role x (or a member of a role that is senior to x) can revoke delegation relationship of a group from any role $y \in Y$, where Y defines the *range of revocation*. Table 3 gives the Can-revoke relation in Figure 2. The first tuple shows that a member of role *HO1* can revoke a delegation relationship of a group from any role in *[Co1, CS]*.

RoleName	Role Range
HO1	[Co1, CS]
Re1	[Re1, AP]

Table 3: Example of can revoke relation.

There are two kinds of revocations (Wang H., Cao J. and Zhang Y. 2003). The first one is weak revocation while the second one is strong revocation. We extend the definition of explicit and implicit members of a role from a user to a group.

Definition 3 A group G is an explicit member of a role x if $(u, x) \in UA$, for all $u \in G$, and that G is an implicit member of role x if for some $x' > x$, $(u, x') \in UA$, for all $u \in G$. ◇

Weak revocation only revokes explicit membership from a user and does not revoke implicit membership. On the other hand, strong revocation requires revocation of both explicit and implicit membership. Strong revocation of $G's$ membership in x requires that G be removed not only from explicit membership in x, but also from explicit (implicit) membership in all roles senior to x. Strong revocation therefore has a cascading effect upwards in the role hierarchy. For example, suppose there are two delegations $((Tony, DIR), (Ahn, AP), Friday)$ and $((John, Re1), (Ahn, AP), Friday)$ and Tony wants to remove the membership of AP from Ahn on Friday. With weak revocation, the first delegation relationship is removed, but the second delegation has not yet removed. It means that Ahn is still a member of AP. With strong revocation two delegation relationships are removed and hence Ahn is not a member of AP.

5 XML implementation

This section presents the implementation of the group delegation with *XML* technology. The format of a group delegation from Section 3 is $((u, r), (G, r), Cons)$. To maintain the relationship between groups, we extend the definition of senior and junior role to the definition of senior and junior group.

Definition 4 A group $G1$ is senior to a group $G2$ if any member of $G1$ has the power of the member in $G2$ and may have additional power but not vice versa. ◇

Let $G1 > G2$ signify that $G1$ is senior to $G2$. Hence a member of $G1$ is considered senior to a member of $G2$. If $G1$ is senior to $G2$ we also say that $G2$ is junior to $G1$. For convenience we use the files *grouphie.xml* and *rolehie.xml* to store the group hierarchy and the role hierarchy of the children and parents groups and roles.

Based on Table 1, a part of the group hierarchy of Figure 2 is modelled in *grouphie.xml* using *IDREF* attibutes (Michael H. 2001) as shown in Table 4. The hierarchy is not a tree but a graph, for clarity and conciseness, Table 4 shows the hierarchy as a nested relation. The first column gives the group name, the second column gives the immediate parent groups of that group, and the third column gives the immediate children. The ϕ means that the group has no parent or child as the case may be. Using *grouphie.xml*, we can find all seniors and juniors for a group by respectively chasing the parents and children using simple *XPath* query expressions. An example of the role hierarchy of Figure 2 is represented in *rolehie.xml* using *IDREF* attibutes as shown in Table 5.

Role Name	Senior role	Junior role
DIR	ϕ	HO1, HO2
HO1	DIR	Co1, Re1
HO2	DIR	Co2, Re2
Co1	HO1	AP
Re1	HO1	AP
AP	Co1, Re1	CS
CS	AP, AsP	ϕ

Table 5: Group hierarchy of Figure 2.

Definition 3 has described a group to be an *explicit* or *implicit* member of a role. A group may be an *explicit* and *implicit* member of a role simultaneously. To simulate a role hierarchy we use information about explicit and implicit membership in *roleDB*. However, *roleDB* is not sufficient to distinguish the case where a group is both an explicit and implicit member of some role from the case where the group is only an implicit member of the role. For this purpose we introduce another file *explicit.xml* that keeps information about explicit membership only.

There is a procedure for delegating a user to a group in our implementation. The procedure call is *Delegate (role, group)*. The parameters role and group specify which role is to be delegated to a group. The delegation function has the following main steps: 1) Select a role to be delegated (or revoked); 2) Select a group to delegate (or revoke); 3) Check whether ro not the group satisfies the prerequisite condition in the relation *Can_deleagte*; 4) Setup constraints; and 5) Update the *DELGR* database. For delegation revocation, instead of the steps 3 and 4, check whether or not the revoked role is in the role range of the relation Can_revoke. The *DELGR* database maintains group hierarchy information (*grouphie.xml*), role hierarchy information (*rolehie.xml*), explicit membership (*explicit.xml*), and the $Can\_delegate$ and Can_revoke relation tables.

In order to make our implementation more convenient we developed a graphical user interface which interacts with this procedure to do role-based group delegation. The graphical user interface is illustrated in Figure 5. This interface was developed using *XUL* and is used to initiate group delegation instead of typing the above procedure call. This implementation is convenient for users since they only need to define the group hierarchy and the relation can-assign.

6 Comparisons

The closed work to this paper is on mobility of user-role assignment (Wang H., Sun L., Zhang Y., and Cao J. 2005) and role-based delegation (Barka E. and Sandhu R. 2000a).

Our previous work (Wang H., Sun L., Zhang Y., and Cao J. 2005) discussed the mobility of user-role relationship in *RBAC* management and provided new authorization allocation algorithms for *RBAC* along with mobility that are based on relational algebra operations. They are the authorization granting algorithm, weak revocation algorithm and strong revocation algorithm. The paper does not use role delegation but instead defines the role mobility, whereby a user with an mobile role may further grant other

Group Name	Parent group	Child group
Tony	ϕ	Project1, Project2
Project1	Tony	Ahn
Project2	Tony	Ahn
Ahn	Project1, Project2	ϕ

Table 4: Group and its roles hierarchy of Figure 2.

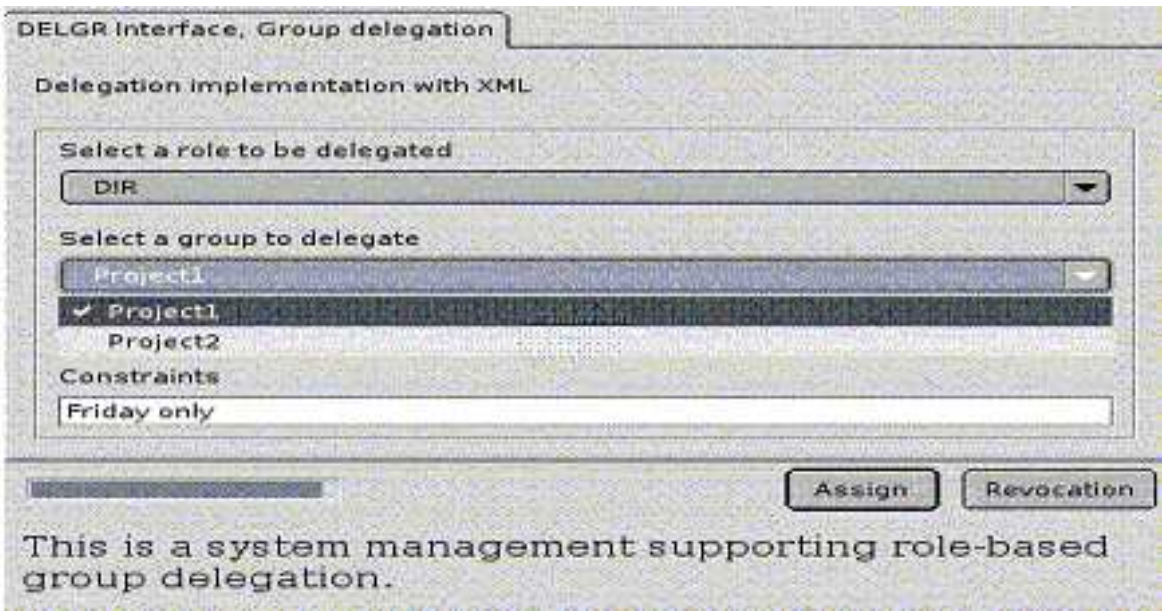

Figure 5: Group delegation interface.

roles but she/he cannot accept other roles if she/he has an immobile role. The mobility could be viewed as a special case of role-based delegation in their work. But some important delegation features such as delegation conflicts and delegation revocation have not been considered. By contrast, the work in this paper provides a rich variety of options that can deal with delegation authorization and revocation.

Barka and Sandhu (Barka E. and Sandhu R. 2000a) proposed a simple model for role-based delegation called *RBDM0* within *RBAC0*, the simplest form of *RBAC*96 (Sandhu R. 1997). They developed a framework for identifying interesting cases that can be used for building role-based delegation models. This is accomplished by identifying the characteristics related to delegation, using these characteristics to generate possible delegation cases. Their work is different from ours in three aspects. First, it focuses on a simple delegation model supporting only flat roles and single step delegation. Some important features such as role hierarchies, constraints and revocations were not supported. By contrast, our work has analysed delegation authorization and revocation models with constraints involving role hierarchies. Second, they neither gave the definition of role-based delegation relation, which is a critical notion to the delegation model nor discussed the relationships among original user and delegated user. By contrast, the delegation framework in this paper is based on original user and delegated user since the delegating relationship in this paper has five components $((u, r), (u', r'), Cons)$. Third, they have not discussed group-based delegation, but we have analysed elements and functions in group delegation as well as its implementation with *XML*.

7 Conclusions

This paper has discussed a role-based delegation model and its implementation with *XML*. We have analysed not only a delegating framework including delegating authorization and revocation with constraints, but also group-based delegation. To provide a practical solution for role-based group delegation, we have analysed role hierarchies and the relationship of senior and junior roles. The theory in this paper was demonstrated by its implementation with *XML*. The work in this paper has significantly extended previous work in several aspects, for example, the group-based delegation, group delegation authorization with prerequisite conditions and revocation authorization.

The future work will be the development of a system management with *XML* which involves the role-based group delegation subsystem.

Acknowledgment

The authors would like to thank reviewers for their good suggestions and comments.

References

Abadi M., Burrows M., Lampson B., and Plotkin G. (1993), 'A calculus for access control in distributed systems', *ACM Trans. Program. Lang. Syst.* **15**(4), 706–734.

Aura T. (1999), 'Distributed access-rights management with delegation certificates', *Security Internet programming* pp. 211 – 235.

Barka E. and Sandhu R. (2000a), Framework for role-based delegation models and some extensions,

in 'Proceedings of the 16 Annual Computer Security Applications Conference', New Orleans, pp. 168–177.

Barka E. and Sandhu R. (2000b), Framework for role-based delegation models, *in* 'Proceedings of the 23rd National Information Systems Security Conference', Baltimore, pp. 101–114.

Bell D.E., La Padula L.J. (1976), 'Secure computer system: Unified exposition and multics interpretation', Technical report ESD-TR-75-306.

Blaze M. Feigenbaum J., Ioannidis J. and Keromytis A. (1999), 'The role of trust management in distributed system security', *Security Internet programming* pp. 185 – 210.

Feinstein H. L. (1995), Final report: Nist small business innovative research (sbir) grant: role based access control: phase 1. technical report, *in* 'SETA Corp.'.

Ferraiolo D. F. and Kuhn D. R. (1992), Role based access control, *in* '15th National Computer Security Conference', ferraiolo92rolebased.html, pp. 554–563.

Li N. and Grosof B. N. (2000), A practically implementation and tractable delegation logic, *in* 'IEEE Symposium on Security and Privacy', pp. 27–42.

Michael H. (2001), *XSLT Programmer's Reference*, Wiley.

Sandhu R. (1997), Rational for the *RBAC*96 family of access control models, *in* 'Proceedings of 1st ACM Workshop on Role-based Access Control', ACM Press, pp. 64–72.

Sandhu R. (1998), Role activation hierarchies, *in* 'Third ACM Workshop on RoleBased Access Control', ACM Press, pp. 33–40.

Wang H., Cao J. and Zhang Y. (2001), A consumer anonymity scalable payment scheme with role based access control, *in* '2nd International Conference on Web Information Systems Engineering (WISE01)', Kyoto, Japan, pp. 53–62.

Wang H., Cao J. and Zhang Y. (2003), Formal authorization allocation approaches for permission-role assignments using relational algebra operations, *in* 'Proceedings of the 14th Australian Database Conference ADC2003', Adelaide, Australia.

Wang H., Cao J., Zhang Y. (2005), 'A flexible payment scheme and its role based access control', *IIEEE Transactions on Knowledge and Data Engineering* **17**(3), 425–436.

Wang H., Sun L., Cao J., and Zhang Y. (2004), Anonymous access scheme for electronic-services , *in* 'Proceedings of the Twenty-Seventh Australasian Computer Science Conference (ACSC2004)', Dunedin, New Zealand, pp. 296–305.

Wang H., Sun L., Zhang Y., and Cao J. (2005), Authorization Algorithms for the Mobility of User-Role Relationship, *in* 'Proceedings of the 28th Australasian Computer Science Conference (ACSC2005)', Newcastle, Australia, pp. 167–176.

Wang H., Zhang Y., Cao J. and Kambayahsi Y. (2004), 'A global ticket-based access scheme for mobile users', *Special Issue on Object-Oriented Client/Server Internet Environments, Information Systems Frontiers* **6**(1), 35–46.

Wang H., Zhang Y., Cao J. and Varadharajan V. (2003), 'Achieving secure and flexible m-services through tickets', *IEEE Transactions on Systems, Man, and Cybernetics, Part A, Special issue on M-Services* pp. 697–708.

Yao W., Moody K. and Bacon J. (2001), A model of oasis role-based access control and its support for active security, *in* 'Proceedings of ACM Symposium on Access Control Models and Technologies', pp. 171–181.

Zhang L., Ahn G., and Chu B. (2001), A rule-based framework for role-based delegation, *in* 'Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT 2001)', Chantilly, VA, pp. 153–162.

Zhang L., Ahn G., and Chu B. (2002), A role-based delegation framework for healthcare information systems, *in* 'Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT 2002)', Monterey, CA, pp. 125–134.