CHAPTER IV

ROLE-BASED GRID DELEGATION MODEL

4.1 Introduction

We discussed two major concerns of grid computing security namely direct authorization and secure inter-operability [91] in the previous chapter. In this chapter we address another equally important aspect of grid security namely, indirect authorization (delegation). Grid delegation is a procedure by which a valid user endows another user or program or a service with the ability to act on that user’s behalf. Delegation is in fact the primary form of authorization in grids. The large and geographically distributed, dynamic, heterogeneous and scalable grid environment poses unique delegation requirements. Presently there are no standard mechanisms to guide grid delegation. As credential delegation (proxy delegation) has its own limitations in a dynamic grid environment, the need for a new conceptual model is felt to effectively formulate the grid delegation requirements. Here, we present a framework called Role- Based Grid Delegation Model (RB-GDM) for delegating access rights in grids. The basic unit of delegation in our model is role. Derived from the standard RBAC formalisms, this framework explores various approaches for authorization and revocation of delegation.

The chapter begins with an introduction to the delegation requirements in grid environment. The rest of the chapter is organized as follows. We first discuss the delegation requirements in grids in section 2. We also present the existing approaches to delegation. In section 3, we propose the role-based grid delegation framework for implementing delegation and revocation in grids. The RB-GDM interconnection framework has been presented in section 4. It includes intra-domain as well as inter-domain topologies. We also extend the delegation model to meet the requirements of delegation and revocation across domains (inter-domain or cross-domain delegation. Finally section 5 summarizes the chapter.
4.2 Delegation Requirements in Grids

Delegation is an important aspect of grid security. In general, delegation is defined as the action whereby an active entity in a system authorizes (delegates) its authority to another entity to execute some functions on its behalf. Delegation in computing systems can take many forms: human to human, human to machine, machine to machine and even machine to human [200]. The consequence of delegation is propagation of access rights [172]. Access rights propagation in a decentralized collaborative system like the grid presents difficult challenges for traditional access control mechanisms and models.

A delegation activity in a grid environment needs to be monitored and controlled in such a manner that the resource inside the domain can stay protected. Though various forms of delegation are possible, the most common delegation activities in present day grids are user to user delegation and user to machine delegation. Most of the resource access in grids takes place through remote authorization and is achieved by user to machine/process delegation. User to machine delegation is the mechanism whereby a user in a distributed environment authorizes a system to access remote resources on his behalf [93]. In some cases the user may delegate the rights to one of the several permissible roles or identities [110]; in order to limit the actions of the process to some subset of the rights that user is authorized to. This form of delegation is generally referred to as limited or restricted delegation.

The support of roles is crucial in an open computing environment like the grids, where not all individuals may be registered with a service and requests may arrive from previously unknown parties. In such a context, the ability to invoke a service often depends on the capacity in which a user can operate rather than on who the user actually is. This capacity is characterized as a role. The concept of roles is different from that of groups in the sense that unlike groups, roles carry a dynamic behavior and can be activated and deactivated by users and service providers as required. Though current grid systems support delegation through impersonation (by issuing a proxy certificate), the revocation of delegation tracing and multi step delegation still remain as the major challenges.

We propose a grid delegation approach built on Role Based Access Control
RBAC has been accepted in the access control design and architecture of security in enterprises as an alternative to traditional discretionary and mandatory access control models. RBAC is designed to suit large-scale enterprise-wide systems and works on the notion that permissions are associated with roles. As explained in Chapter I of the thesis, the users are assigned to appropriate roles and users acquire permissions by being members of roles. Roles can be granted new permissions and can be easily revoked from roles as and when needed. Thus RBAC provides a way to empower individual users and service providers through role-based delegation \cite{120} in a fully distributed environment like grids.

4.2.1 Existing Delegation Approaches

The core theme behind delegation is that of delegating rights, obligations, privileges and authority. The basic idea of role-based delegation is that users themselves may delegate role authorities (rights) to others to carry out some functions authorized to the former \cite{45}. Related delegation approaches like \cite{82}, \cite{83} generally assumes that the rights to be delegated have already been decided by a separate mechanism, and it is just a matter of encoding and checking these rights. The general need for delegation is exemplified by the Digital Distributed System Security Architecture, a comprehensive collection of security services and mechanisms for general-purpose distributed systems \cite{27}. Gasser.M and McDermott.E \cite{157} discuss various issues on implementing delegation in the distributed environments. Target restrictions and delegate restrictions are the common forms of delegation-related restrictions. Calvelli et.al \cite{25} analyzed the use of delegation protocols for distributed systems. \cite{63} suggested a weighted graph approach to authorization delegation and conflict resolution.

Delegation of rights is a significant problem in middleware like Globus. Grid Security Infrastructure, the security component of Globus supports an interface for delegation. But, it is based on credentials and proxies. The reliance on GSS-API in Globus hinders efforts at delegation in Globus, because GSS-API does not offer a clear solution to delegation and impedes restricted delegation. Delegating the user’s full rights could result in impersonation. There have been some efforts in Globus to adopt and support the Generic Authorization and Access-Control API
However, it does not propose any systematic procedure by which we can determine the rights that should be delegated. Mehran et al in [101] suggested a delegation protocol for grid computing systems.

The study conducted on the existing delegation attempts lead to the conclusion that delegation, in its present form is not effective to meet the grid authorization requirements. We, therefore, present a delegation model for grid resource access based upon roles through which only the rights assigned to roles can be delegated, thereby making it fine-grained. Unlike the existing identity-based approaches, this framework treats grid delegation as an authorization problem.

4.3 Grid Delegation Framework

In this section, we present a grid delegation approach called the Role-Based Grid Delegation Model (RB-GDM). This model is aimed at supporting role-based delegation and revocation. Our work is built on the following reference models: RBAC96 model [155], the RBDM0 model [45], RDM2000- A Rule-Based Framework for Role-Based Delegation and Revocation [200], Rule-Based Delegation Model 1 (RBDM1) [46] and RB-GACA: A RBAC Based Grid Access Control Architecture [202].

4.3.1 Basic Elements from Reference Models

The Role Based Access Control Model (RBAC) has the following basic elements: users \( U \), roles \( R \) and permissions \( P \). Roles are assigned to individual users and permissions are assigned to roles. According to this model, a user is an individual; a role is a job function indicative of the responsibility and permission indicates the approval to execute some method or perform some action. We generalize the term user to make it represent an individual, an intelligent autonomous agent, or a machine in the grid environment. We consider the end user of a grid resource as the basic user or original user. There are two sets of users associated with a role \( r \): as given in.

- Original users: Those users who are directly assigned roles
- Delegated users: Those users who are indirectly assigned (delegated) the roles by original users
Constraints on User to User Role Delegation

1. At any time a user can have its original role and the roles which it has acquired through delegation by other users

2. A user is authorized to delegate its original role only or the roles it has acquired through delegation. It is meaningless for a user to delegate other users’ roles

3. Roles follow a hierarchy. Thus a user with lower role is not authorized to delegate a role to a user of higher role

4. It is meaningless for a user to delegate a particular role to another user to which he already belongs

5. A user of higher role has privileges to delegate its junior roles also. This enforces the principle of least privilege

6. A user will not be able to delegate a role which has no further delegation allowed

Each constituent domain of the grid virtual domain (VO) may have its own role hierarchies. As mentioned in Chapter I, a role hierarchy is a structure reflecting an organization’s lines of authority and responsibility and is represented using a partial ordering operator \( \geq \). For example, if role1 \( \geq \) role2, then role1 inherits permissions of role2. Though role hierarchies [46] in different local domains vary in their semantics, they are required to enforce the “Principle of Least Privilege” [17] which states that in a particular abstraction layer of a computing environment, every module (such as a process, a user or a program on the basis of the layer we are considering) must be able to access only such information and resources that are necessary to its legitimate purpose. The fundamental components of RBAC and RBDM0 which form the basis for our model (named RB-GDM) are depicted as follows in Table 3 and Table 4 respectively. These reference models have been detailed in [155] and [45].
4.3.2 Role-Based Delegation Model

The delegation model RBDM0 [45] suggests only user-to-user delegation, which is the simplest form of delegation. Our model is grid specific. As mentioned earlier, the end-user is considered as the basic user. The end-users are geographically distributed, can be dynamic and may get disconnected from the virtual organization (VO) after the submission of a job. Therefore, end-user-to-end-user delegation is not possible grids. As the end user cannot directly delegate an entity, we need to look at other forms like user-to-machine, machine-to-machine or user-to-process delegation. The basic idea of delegation is to decide who is delegating or revoking what to whom. Thus the entities in delegation are the grantor, the grantee and the rights granted [182]. To suit the requirements of a grid environment and to distinguish delegation from direct authorization, we call the grantor of delegation as a delegator, the grantee as delegatee and the rights that are granted through the roles as delegated rights.

As discussed in Chapter I, the RBAC role hierarchy includes flat roles and hierarchical roles. For flat roles, no inheritance of permissions between roles is involved. With regard to delegation, we define roles as either further delegatable or non-delegatable based on whether a subset of roles can further be delegated or
The first challenge in framing a grid delegation model is to identify the roles assigned to the users in the grid environment. The subject or active entity in a grid transaction can be:

- a user (an individual)
- a process or
- a machine

Among users, there can be:

- administrators (who possess all the administrative and super user privileges - eg, a domain administrator)
- basic users (who can submit large scale jobs, access resources etc)
- developers (whose main objective is to handle the programming and middleware aspects) or
- guest users (who can only browse the information or submit very small scale jobs)

The components that constitute a grid delegation mechanism include the delegator, the delegatee, the delegated rights and the associated constraints. Multi-step delegation is possible with delegatable roles whereas single step delegation applies to roles which are non-delegatable. Sometimes the delegator grants full permissions to the delegatee. We call this as total delegation. If the delegator assigns only a subset of the rights to the delegatee, it is called partial delegation. We make the following design criteria in RB-GDM.

- A role is considered to be the basic unit of delegation and is a job function
- The term user in our model represents an end-user, a process or a machine in the broader perspective
- Delegation between entities in the same role is not allowed since the same roles will have same permissions
Both single step delegation and multi-step delegation are permissible
Delegation can be granted in total or can be partial
Each delegating role \( r \) has two types of members namely the original members \( O_r \) and delegated members \( D_r \)
Grid user can be a domain administrator, an advanced user, a basic user or a guest user

Before we build a model, it is essential to define the components which constitute the model. Therefore we present the following definitions 1 to 4 which are detailed in Table 5, Table 6, Table 7 and Table 8.

1. In Definition 1, the basic components like grid domain, users, roles, permissions and the user-role, role-permission assignments which are required to build the Role-Based Grid Delegation Model (RB-GDM) are explained
2. Definition 2 explains how the access rights can be delegated through authorization.
3. Definition 3 shows as to how we can give different role permissions like delegable and nondelegable
4. Definition 4 depicts the meaning of “revocation of delegation”

The interaction of these components facilitates a delegation mechanism as per the algorithms for authorization and revocation of delegation. Algorithm 3 gives the method for authorization of delegation as per Definition 2. Algorithm 4, 5 and 6 show the methods for revoking the delegated rights by various means such as grant-dependent, grant-independent or time-out. First, we present the intra-domain (single domain) delegation procedure and in the next subsection, we show as to how we can perform delegation across domains.

There are two possible ways in which revocation can be done. The first approach is by revocation using time out (attaching a time constraint to every assigned delegation). Timeout revocation though self triggering, is not enough to ensure secured delegation. For tasks, which take much longer periods to complete, this method is not suitable. Tracking the delegator is another issue with this type
Table 5: Definition 1 RB-GDM Components

<table>
<thead>
<tr>
<th>Domain</th>
<th>Users</th>
<th>Roles</th>
<th>Permissions</th>
<th>User-Role Assignment</th>
<th>Permission-Role Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_l</td>
<td>U_l</td>
<td>R_l</td>
<td>P_l</td>
<td>UA_l ⊆ U_l × R_l</td>
<td>PA_l ⊆ P_l × R_l</td>
</tr>
</tbody>
</table>

Table 6: Definition 2 RB-GDM Delegation

RB-GDM controls role-based delegation by means of predicate can_delegate ⊆ R_l × R_l* between two domains of a grid G. It is a non-reflexive relation since the user in a role cannot delegate his membership to another user in the same role. The meaning of the above relation is that if u_l1 ∈ R_l and u_l2 ∈ R_l, then (r_l1, r_l2) ∈ can_delegate, i.e., (u_l2, r_l1) ∈ UAD_l, provided the constraints are satisfied.

Table 7: Definition 3 RB-GDM Role-Permissions

Each role can have two types of permissions
- Delegatable permissions PD_l
- Non-delegatable permissions PN_l

From the RBDM0 model, we extend the following permission components for RB-GDM, where P_l is a set of regular permissions
- PA_l = PDA_l ∪ PNA_l, where PA_l, PDA_l and PNA_l are ⊆ P_l × R_l

which include many to many role to permission assignment relation, many to many delegatable role to permission assignment relation and many to many non-delegatable role to permission assignment relation respectively.
- P_l(r_lj) = {p_lk | (p_lk, r_lj) ∈ PA_l}

Similarly for PDA_l(r_lj) and PNA_l(r_lj) as well
of delegation and revocation. These issues can be handled by explicit revocation (either grant-dependent or grant independent).

**Table 8:** Definition 4 RB-GDM Revocation

<table>
<thead>
<tr>
<th>The revocation procedure can be through any of the following ways</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Remove one or more pieces of permissions from delegation</td>
</tr>
<tr>
<td>• Revoke delegation role owned by a fixed delegable role</td>
</tr>
<tr>
<td>• Remove one or more pieces of permissions from a fixed delegable role to its regular (original) role</td>
</tr>
</tbody>
</table>

The procedure for role-based delegation is shown in algorithm 3 below. It shows how the permission is associated with the delegated role and how delegation is granted or denied. Revocation of delegation is as important and crucial as authorization of delegation. The revocation procedure (Grant Dependent) is done as represented in the algorithm 4.

---

**Algorithm 3 Role-Based Delegation**

**Input:**
1. A delegation request \((u_i^l, r_j^l)\)
2. \(UA_l\)
3. \(PA_l\)

**Output:**
- True, if the delegation is granted
- False, if delegation is denied

**Begin**

**Step 1:** \(U^l(r_j^l) \leftarrow \text{get}(UA_l); \) /* from Definition 2, returns the user-role assignment set for the domain*/ \(UA_l\) is such that \(UA_l = UAO_l \cup UAD_l\)

**Step 2:** for \(U^l(r_j^l) = \{u_i^l,(u_i^l, r_j^l) \in UA_l\}\)

- if \(P^l(r_j^l) = \{p_l^k,(p_l^k, r_j^l) \in PDA_l\} \) /* returns true if the permission is associated with the delegated role and delegation is granted, otherwise the condition returns false and delegation is denied. This decision is guided by the policies of the role-permission relationship set of the given domain*/

  if \(((u_i^l,r_j^l) \in \text{can\_delegate})\)

  **Step 3:** return true;

  **Step 4:** return false;

**End**

/*Example : Let the delegated user be in a role “basic user”. It sends a request for delegation along with the role and access specifications. The delegator for example, an administrator maps the user-role relations and the role-permission relations and grants or rejects delegation accordingly.*/

---

81
**Algorithm 4** Revocation of Delegation-Grant Dependent

**Input:**

\[1 \] \( UA_l \)

**Output:**

- True, if the delegation is revoked
- False, if delegation is not revoked

**Begin**

\[ \text{if} \ ( (u_i, r_j) \in can\_revoke) \]

- **Step 1:** return true;

\[ \text{else} \]

- **Step 2:** return false;

**End**

/* Example : Let the delegatee user be a process and the delegator be an administrator. The execution of the algorithm results in revocation of delegated rights by the delegator (in this case the administrator) himself.*/

Algorithm 5 shows the steps by which grant-independent delegation is revoked. Algorithm 6 gives the methodology for delegation revocation based on timeout concept.

**Algorithm 5** Revocation of Delegation-Grant Independent

**Input:**

\[1 \] \( UA_l \)

**Output:**

- True, if the delegation is revoked
- False, if delegation is not revoked

**Begin**

\[ \text{if} \ ( (u_i', r_j') \in can\_revoke) \]

- **Step 1:** return true;

\[ \text{else} \]

- **Step 2:** return false;

**End**

/* Example : Let us consider a user \( \{u_i\} \) in a role “basic user” and the delegator be another basic user who is granted rights by a site administrator. When this algorithm is invoked, it matches the user (delegatee) with its corresponding role i.e., guest and the delegated rights are revoked by the administrator and not by the delegator.*/

**4.4 RB-GDM Interconnection Framework**

We propose three approaches, one for intra-domain access control and the remaining two for inter-domain access control [167]. These interconnection networks are implemented using realized based on the definitions 1 to 4 given in section 4.3.2.
The interactions between the components of the RB-GDM are governed by the algorithms. The basic procedure to authorize and revoke delegation is the same for all the three interconnection networks except that RB-GDM components will be organized in different forms in each case.

### 4.4.1 Intra-Domain Role Based Delegation

It is essentially similar to an enterprise wide role delegation where the entities within a domain delegate tasks to others using a hierarchical relationship. The appropriate association would be that of the superior entity delegating to the subordinate. Figure 17 illustrates this framework. The RB-GDM core model that supports intra-domain role-based delegation is shown in Figure 18.

### 4.4.2 Inter-Domain Role Based Delegation

Role-based delegation across domains can be done mainly in two ways; by treating the two domains as equals (peers) in the association or by considering a master-slave relationship.

#### 4.4.2.1 Peer-to-Peer Role-Based Delegation

This approach is suitable when transactions take place between two domains of flat association i.e. there exists no hierarchical relationship (for example, between two
Figure 17: Intra-Domain Role-Delegation Framework

Figure 18: RB-GDM Core Model
investment banking domains). This type of delegation is based on the mapping between a role in the delegator domain to its corresponding equivalent role in the delegatee domain both being peer roles. Figure 19 represents the peer-to-peer interconnection framework.

**Figure 19:** Peer to Peer Interconnection Framework

The RB-GDM (Role-Based Grid Delegation Model) for peer-to-peer set up is as in Figure 20.

**Figure 20:** Peer-to-Peer Inter Domain RB-GDM

### 4.4.2.2 Master/Slave Role Based Delegation

This approach helps to grant role delegation between two domains, which have a master/slave kind of association between them. It implies that the roles of one domain act as superiors to the roles of another domain. An example of such an arrangement is the association between a central bank domain and a subsidiary bank domain (hierarchy between two different domains). Typical hierarchical domain relationship can also exist in a grid between a virtual organization (VO) or
virtual domain and the member domains of the VO.

Figure 21 shows the hierarchy topology for inter-domain role delegation.

![Hierarchy Topology of Inter Domain Role Delegation](image)

**Figure 21:** Hierarchy Topology of Inter Domain Role Delegation

The RB-GDM for master/slave inter domain role based delegation is as in Figure 22.

![Master/Slave Inter-Domain RB-GDM](image)

**Figure 22:** Master/Slave Inter-Domain RB-GDM

### 4.4.3 Inter-Domain/Multi-Domain Role Delegation Model

We propose a grid delegation model for delegation across domains. Jajodia et.al [84] proposed a generic model for multi-domain delegation. Our model is based on the role-equivalence established between roles of different domains. We rank
the roles using the role mapping architecture [168] which we proposed in Chapter III. When a request comes from a user in one domain for a resource in another domain, we derive the user’s equivalent role in the second domain. Assume the grid user user1 has a role a3 as shown in Figure 14. He wants to access a resource which requires a minimum role of b4 in the hierarchy. Since the global rank of a3 is more than that of b4, user1 will be granted access. At the same time he will be given a temporary role of b2 in the domain B.b. Our model takes care that the global rank of the temporary assigned role is closely equal to the global rank of the requested role. Now in the above mentioned scenario the user ‘user1’ will be assigned a temporary role of b2 (suppose) in the domain B.b. Figure 23 shows the sequence diagram for cross-domain role-mapping.

![Sequence Diagram for Cross Domain Role Mapping](image)

**Figure 23**: Sequence Diagram for Cross Domain Role Mapping

Figure 24 depicts delegation across different domains of a grid computing environment.

Algorithm 7 shows the steps through which a user is delegated across another domain.
Algorithm 7 AcceptDelegate

<table>
<thead>
<tr>
<th>Input: User Token</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: Delegation deny/Grant</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>Authorize User</td>
</tr>
<tr>
<td>Create user with username and other credentials</td>
</tr>
<tr>
<td>Add user to the Temporary user set</td>
</tr>
<tr>
<td>retrieve the role whose global rating is closest-to that of user and assign user with that role</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

4.4.3.1 Revocation of Rights-Cross Domain Environment

The rights once granted, should not be held for long by the delegate. Such a scenario may invite misuse. It is necessary to have a robust access control mechanism which encompasses a strong revocation model to revoke the rights passed on by a user. Earlier, in the chapter, we presented an algorithm for revocation of intra-domain or single domain delegation. Here, we incorporate revocation in the cross-domain environment. In a scenario where a user, user1 wants to revoke all his assigned roles, then the user invokes the doRevoke method of the domain B_b. It recursively calls the same method from the domains where it has further delegated the user’s rights. It then stops all the jobs which are running with that user’s rights, then removes the user from the temporary users’ set and then returns the token. Thus the revocation of user’s rights happens recursively in the above mentioned fashion as shown in Figure 25.

The algorithm for the recursive revocation procedure is as shown in the Algorithm 8.
Revocation of delegated rights in a multi-domain environment can be done through any of the following means.

- **Revocation-Grant Dependent**: In this model, the delegate can revoke any right that he has delegated at any point of time. This means that he can issue revocation at any time to the Authorization server of the domain to which the right has been delegated.

- **Revocation- Time Dependent**: In this model the delegated right expires after a stipulated amount of time, when it is revoked automatically in the delegated domain according to the revoke algorithm as given above.

- **Revocation- Administrator Dependent**: In this model, the administrator of the delegate domain or the domain in which the role has been delegated can revoke the role. We define a delegation path as “a set of domains to which the role has been delegated from one domain to another in order”. This path would contain the global rank of the role delegated in that particular domain with the names of the domain.
4.5 Chapter Summary

The main contribution of this chapter is the proposed Role-Based Grid Delegation Model (RB-GDM). The model is based on the standard RBAC and RBDM models and provides a flexible indirect authorization mechanism for grids. This framework is an innovative approach towards delegation in grid environment, as it uses a role as the criterion instead of a proxy credential. By categorizing the grid users into various roles and delegating access rights through roles, we also achieve granular access control. We provided three different inter-connection frameworks to depict the delegation scenarios in the intra-domain and inter-domain grid setup. Our model supports both peer to peer as well as hierarchical role relationships. Delegation based on role hierarchies ensures restricted access rights to the resources. The proposed model also provides mechanisms for revocation of delegated rights. RB-GDM is a common framework to support single domain as well as multi-domain grid delegation and revocation.