Role-Based Delegation Model/ Hierarchical Roles (RBDM1)

Ezedin Barka
College of Information Technology
University of the United Arab Emirates
Al Ain, United Arab Emirates
ebarka@uaeu.ac.ae
And
Ravi Sandhu
Laboratory for Information Security Technology
Information and Software Engineering Department, MS 4A4
George Mason University, Fairfax, VA 22030, USA
Sandhu@gmu.edu

Abstract

The basic idea behind delegation is that some active entity in a system delegates authority to another active entity in order to carry out some functions on behalf of the former. User delegation in RBAC is the ability of one user (called the delegating user) who is a member of the delegated role to authorize another user (called the delegate user) to become a member of the delegated role. This paper introduces a new model, which we consider it to be an extension of RBDM0 [BS2000].

The central contribution of this paper is to introduce a new model, referred to as RBDM1 (Role-Based Delegation Model/ Hierarchical Roles), that uses the details from RBDM0, which was described in the literature by barka and Sandhu [BS2000] to address the temporary delegation based on hierarchical roles. We formally defined a role-based delegation model based on hierarchical relationship between the roles involved. We also identified the different semantics that impact the can-delegate relation, we analyzed these semantics to determine which ones we consider as more appropriate in business today, thus allowed in our model, and provided a justification to why those selections are made

1. Introduction

This paper describes the ways by which RBDM0 is extended to address more complicated issues that come along with hierarchical roles.

Hierarchies are natural means for structuring roles to reflect an organization's lines of authority and responsibility (figure 1). By convention, more powerful (senior) roles are shown toward the top of these diagrams, and less powerful (junior) roles toward the bottom. In figure 1.a, the junior-most role is that of the health-care provider. The physician role is senior health-provider and thereby inherits all permissions from health-care provider. The physician role can have permissions besides those it inherited. Permission inheritance is transitive. So, for example, in figure 1.a, the primary-care physician role inherits permissions from both the physician and health-care provider roles. The primary-care physician and the specialist physician both inherit permissions from the physician role, but each will have different permissions directly assigned to it. Figure 1.b illustrates multiple inheritances of permissions, where the project supervisor role inherits from both the test engineer and the programmer role.

Mathematically, these hierarchies are partial order. A partial order is a reflexive, transitive, and anti-symmetric relation. Inheritance is reflexive because a role inherits its own permissions, transitivity is a natural requirement in this context, and anti-symmetry rules out roles that inherit from one another and would therefore be redundant.

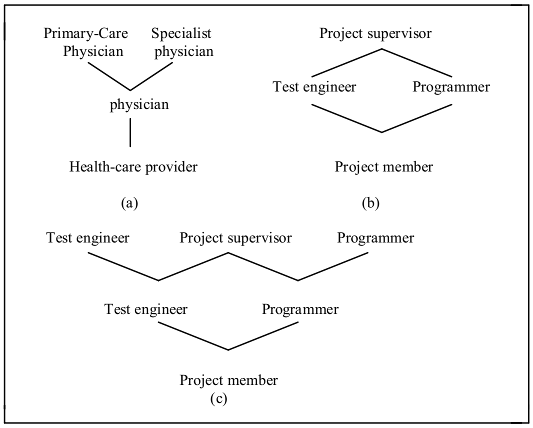

Figure 1. Example of Role Hierarchy

When we extend RBDM0 model to capture the role-to-role delegation using hierarchical roles, we add more complexity to the flat roles model. Here, we have to deal with different kinds of delegations, some of which are not very useful, and some which carry more risk than others.

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

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

The needs above can be justified by explaining delegation as a particular mechanism for collaborative working. Suppose a group of employees need to work together. In delegation, the members of the group do not work in tandem; their rights are used by delegates of the group without their participation. This results in a need for trust between members. This trust can be limited in scope by limiting the rights contributed by delegator to delegate.

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

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

The rest of this paper is organized as following: Section 2 provides assumptions and basic elements that are specific to the role-based delegation models in hierarchical roles. Section 3 discusses delegation in RBDM1, and analyzes the deferent semantics of delegation in hierarchical roles. This is addressed in the sub-section 3.1 Section 4 addresses revocation of delegation within RBDM1. Finally, Section 5 provides a summary of the RBDM1 model.

2. Assumptions and Basic Elements

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

- Delegation can only be either downward or cross. Upward is useless because senior roles inherit all the permissions of their junior roles.
- Downward delegation means that a user who is an original member of a role delegates his role to other users who are original members of roles that are junior to the delegation role.
- Cross delegation means that the delegation takes place between users who are members of incomparable roles. For example, a manager in the sales department can delegate his role membership to an auditor from the auditing department in order to do auditing on the sales department.

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

Original members of senior roles are also original members of the roles that are junior to their roles, and delegate members of senior roles are also delegate members of the roles that are junior to their roles. However, this type of membership is considered an implicit membership.

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

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

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

The following section formally defines the role-based delegation model in hierarchical roles**:**

To flow the natural progression from RBAC to RBDM1, we refer to the definitions of RBAC96 and RBDM0 listed below:

Definition 1: The following is a list of the original RBAC96 components:

- U and R and P are sets of Users, Roles, and Permissions, respectively.
- $UA \subseteq U \times R$ is a Many to Many, User to Role assignment relation
- $PA \subseteq P \times R$ is a Many to Many, Permission to Role assignment relation
- Users: $R \rightarrow 2^U$ is a function derived from UA mapping each role r to a set of users where Users(r) $= \{U \mid (U, r) \in UA\}$
- Permissions: $R \rightarrow 2^P$ is a function derived from PA mapping each role to a set of permissions where Permissions (p) $= \{P \mid (P, r) \in PA\}$

Definition 2: The RBDM0 model adds the following components:

- $UAO \subseteq U \times R$ 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

Definition 3: The following is a formal definition of RBDM1:

The definition of RBDM1 is the same as RBDM0, with the following elements added (see figure 2):

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

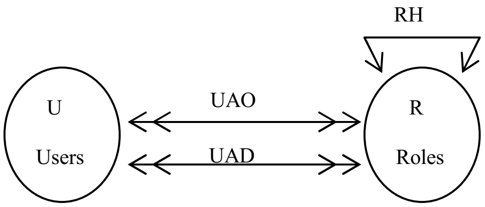

Figure 2: RBDM1

3. Delegation in RBDM1

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

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

In this section, we address how the semantics of delegation in RBDM1 impact the can-delegate relation. We list a number of semantics for the can-delegate relation in RBDM1, we analyze these semantics and identify the ones that make more sense for business today, thus allowed by our model, and we justify our selections by giving some examples. Furthermore, in this section, we address how revocation is handled under the new conditions.

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

The following is a formal definition of an implicit membership:

Definition 4: Let us a say a user U is an explicit member of role x if $(U, x) \in UA$, then a user (U) is considered to be an implicit member of x if for some $x' > x$, $(U, x') \in UA$

Definition 5: The user-role assignment is authorized in RBDM1 by the following relation: Can-delegate $\subseteq R \times R$

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

3.1 Semantics of Delegation in RBDM1

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

In this section, we list and analyze the different semantics that impact delegation in RBDM1 and explained the approach our model takes towards allowing the appropriate semantic of delegation.

Figure 3 depicts organizational role hierarchy and users' role memberships. To illustrate the different semantics of delegation in RBDM1, we use this example in the rest of this section.

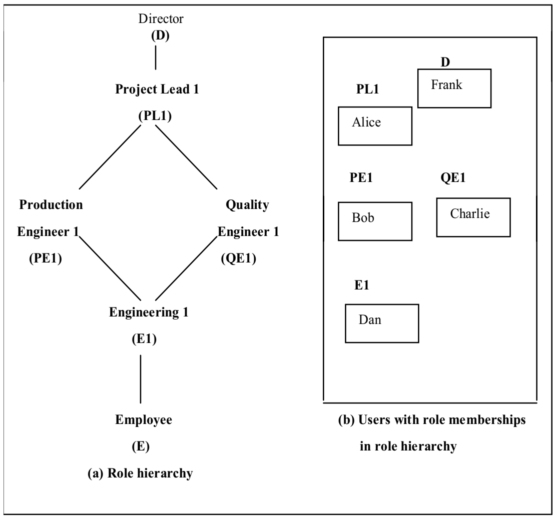

Figure 3: An Example of Organizational Role Hierarchy and Its Users

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

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

2) For $x > y \Rightarrow (y, x) \notin$ can-delegate
$(x, y) \notin$ can-delegate means that a member of a role cannot delegate his role membership to another user who is a member of another role senior to his role. For example, in Figure 3, Alice who is a member of (PL1), cannot delegate PL1 to Frank who is a member of the role director, because by definition, Frank inherits the permission of role PL1.
This semantic is very useful, because it prevents the delegation from being upward.

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

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

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

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

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

For the sake of illustration we use Table 1, in conjunction with Figure 3, to describe the derived semantics of the can-delegate relation. We used all possibilities that result from testing the delegating role/delegated role memberships at any given time.

As the case in RBAC96 and RBDM0, in RBDM1, delegating role members and delegated role members are assumed to be original members. Moreover, through out this discussion, we assumed that all the members shown in figure 3 to be original-explicit members.

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

In the table below, we list all different semantics that resulted form the above conditions.

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

Table 1: Examples of Authorization Functions

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

With the assumption that (PL1, E1) ∈ Can-delegate, the following semantics were derived:

1. In RBDM0, where the relation between roles is flat, the can-delegate relation has very clear meaning: both the delegating and the delegated roles are original/explicit. Therefore, the can-delegate relation has one meaning: (PL1, E1) ∈ Can-delegate. This means that any member of PL1can-delegate to any member of E1 , and vice versa.

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

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

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

In the second scenario, where the delegating role is an original/explicit and the delegated role is an original/implicit (OE, OI), our table shows that because Dan is an implicit member of E1, he is also an explicit member of PE1 and explicit member of QE1. This means that, in addition to being able to delegate PL1 to Dan, Alice can delegate PL1 to Bob, and Alice can delegate PL1 to Charlie. This also means that, Alice can-delegate PE1 to Charlie, and Alice can-delegate QE1 to Bob.

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

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

In conclusion, in this model, we have chosen the most liberal approach of authorizing delegation between users in different roles. This means that our model allows all semantics of the can-delegate relation. This is motivated by the fact that by allowing one semantic or the other will produce anomalies. For example, by allowing only (OE, OE) means that Alice will not be able to delegate PL1 to Bob, and to delegate PL1 to Charlie. However, Alice is allowed to delegate the same role to Dan, which is a less powerful role than that of Bob and of Charlie. Also, by allowing only (OE, OE) will prevent Frank from delegating PL1 to Dan. This is not desirable, because Frank is the most senior role, thus, inherits permission of all other junior roles. Hence, should be allowed to delegate PL1 to anywhere Alice can.

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

The above semantics of delegation are a result of having an active/full hierarchy. If the hierarchy is empty, or collapsed, our model becomes flat and our can-delegate becomes the same as in RBDM0.

4. Revocation in RBDM1

We now turn our attention to the revocation part of RBDM1. Revocation in RBDM1 takes the approach of the classical discretionary access control where the source of the delegation (explicit or implicit) and the identity of the revoker are taken into account in interpreting the revoke operation.

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

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

4.1. Revocation Using Time Outs

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

4.2. Human Revocation

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

- Only the delegator can revoke:

Only the delegating original can revoke. This approach has some advantages and disadvantages. Among the advantages are:

- It gives power to the original delegating member to track and control the behavior of the temporary delegate member.

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

Among the disadvantages of this approach are:

- Protection of the system resources from the delegate member depends solely on the delegating role member. If the delegate member behaves badly in the delegated role, then only the delegating user can revoke his membership, which could take a long time before the delegation can timeout. Allowing any of the original role members to revoke can help mitigate the risk resulting from such situations.

This revocation approach raises some issues that introduce complexity and subtlety. The following discussion addresses these issues.
For the sake of illustration we used Table 1, in conjunction with Figure 3, to discuss the revocation issues associated with the delegation in hierarchical roles.

Suppose that Alice, who is an original member of role PL1 (Alice $\in$ User_O(PL1)), delegates her membership to Bob who is an original member of role PE1 (Bob $\in$ User_O(PE1)), (PE1$\leq$ PL1). Thereby ((Bob, PL1) $\in$ UADE), and ((Bob, r') $\in$ UADI), where, r' is any role that is junior to PL1 (PL1 $\geq$ r'). This is done at Alice's discretion because Alice acts as an owner of role PL1 because of her original membership in that role. Alice can later revoke Bob's delegate membership of role PL1 (and from any role that is junior to PL1). Note that, in this case, a member of any role that is senior to role PL1 cannot revoke Bob's membership in PL1. This is because that senior role is not the actual delegator of role PL1 to Bob. In our example, this means Frank cannot remove Bob from PL1.

Now suppose that Bob was made a member of role PL1 by Alice, and by Dave, who is another member of PL1, not shown in figure 3. If Alice revokes Bob's membership in PL1, then Bob should still continue to retain his membership in PL1, via Dave. Bob can be totally revoked from PL1 only if both Alice and Dave revoke his membership in PL1.

- Cascading Revocation

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

Our model supports the cascading revocation. In the above example, suppose that Alice's membership of role PL1is revoked by a security officer. This will result in the automatic revocation of Bob's membership in role PL1 (and from any roles junior to PL1). Also, if Bob loses his membership in his original role (PE1), this will lead to losing his delegate membership of role PL1 (and any roles junior to role PL1). However, if Dave's membership in role PL1 was in turn given by Alice, then if Alice revokes Bob's membership of PL1, Bob will also lose his membership in role PL1 obtained from Dave. Alice can also revoke the membership of Bob in role PL1 indirectly by revoking Dave's membership of PL1.

- Multiple sponsoring / supporting roles

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

Multiple sponsoring roles is when a user becomes a delegate member in a role by more then one original member in that role. This is also allowed in our model.

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

Definition 6: The role-role revocation is authorized in RBDM1 using the following relation:

Can-Revoke $\subseteq R \times R$

The meaning of can-revoke $(x, y) \in$ can-revoke is that the delegating member of role x (explicit or implicit) can revoke the membership of the delegate member y or any subsets of y in the role x. For example, Alice, who can delegate PE1 to Dan, thereby ((Dan, PE1) $\in$ UADE), can also revoke Dan from PE1, and any roles junior to PE1.

Strong Revocation vs. Weak Revocation

In RBDM1, revocation has impact only on explicit membership and it is strong. Strong revocation requires revocation of both explicit and implicit memberships. A user who is strongly revoked from a role will also be weakly revoked from all roles junior to that role. Strong revocation therefore has a cascading effect downward in the role hierarchy. In weak revocation, a user may be revoked explicitly from a role but continue to maintain an implicit membership in the same role. This situation does not apply in RBDM1 (as shown in examples above) because the delegation was done at the delegator's full discretion. Thus, when he revokes, every related delegation gets revoked.

5. Summary of the RBDM1

In this paper we described the motivation, intuition, and formal definition of a new simple and a non-trivial model for human-to-human delegation using roles called RBDM1 (Role-Based Delegation Model/ Hierarchical Roles) that is based on the Role-Based Access control (RBAC96) developed by [SCFY96]. This new model is considered an extension to the RBDM0, which was a delegation model using flat roles. In this paper we also identified the different semantics that impact the can-delegate relation, we analyzed these semantics to determine which ones we consider as more appropriate in business today, thus allowed in our model, and provided a justification to why those selections are made. We concluded this paper with an explanation of how our model handles the revocation of the previously delegated memberships. Our model has two approaches to implement revocation of previously delegated roles. In the first approach, it appends a lifetime to each delegation. Once that time expires, so does the delegation. The second approach our model uses to implement revocation is allowing users to revoke the memberships of delegated roles (human revocation).

6. References

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

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

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

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

[SB97] Ravi Sandhu and Venkata Bhamidipati. Role-based administration of user-role assignment: The UR97 model and its Oracle implementation. In Proceedings of IFIP WG11.3 Workshop on Data Security. August, 1997.

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

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

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