ARBAC07: A Role-based Administration Model for RBAC with Hybrid Hierarchy

Yue Zhang
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA, USA
zysxqn@cs.pitt.edu

James B.D. Joshi
School of Information Science
University of Pittsburgh
Pittsburgh, PA, USA
jjoshi@sis.pitt.edu

Abstract

Recently, administration of RBAC systems using role-based approach has become very appealing because of the benefits that a role-based approach typically brings. This approach uses RBAC itself to manage RBAC policies so that the administration functions can be decentralized and made more efficient. ARBAC97, ARBAC99, and ARBAC02 are series of well-known solutions for decentralized RBAC administration. However, none of these can be used for RBAC systems that support hybrid hierarchies, which have been shown to be necessary to specify fine-grained RBAC policies. In this paper, we propose the ARBAC07 model based on the ARBAC97, ARBAC99 and ARBAC02 models for an RBAC system with hybrid hierarchy. We show that our model keeps all the advantages of the original model and can further deal with more fine-grained RBAC policies where hybrid hierarchy is needed.

1. Introduction

Role Based Access Control (RBAC) has become widely accepted as a promising alternative to the traditional discretionary access control (DAC) and mandatory access control (MAC) approaches [3, 4, 5, 12]. In RBAC, permissions are assigned to roles and users are made members of roles. An RBAC approach is policy-neutral and flexible. Users can be easily reassigned from one role to another if needed, and roles can also be granted new permissions or existing permissions can be easily reassigned if the function of a role changes.

A crucial challenge for RBAC systems is the development of an efficient enterprise policy administration framework. In modern large enterprise-wide systems, there could be many roles and many more users/permissions [14]. The relationships among the roles, users, and permissions change continuously. Centralized management of such large number of roles, users, permissions and their interrelationships has several drawbacks [11]. Hence, decentralizing the administration of RBAC policies without losing the central control is a challenging goal for system designers and developers.

The use of role itself to manage the RBAC system has become an appealing idea. Sandhu *et al.* [14] have proposed an ARBAC97 (Administration RBAC '97) model consisting of URA97 (User-Role Assignment '97), PRA97 (Permission-Role Assignment '97), and RRA97 (Role-Role Assignment '97) models, which use RBAC to manage RBAC policies. They have further extended this model to ARBAC99 where they separate users/permissions into mobile and immobile users/permissions [15], and later to ARBAC02, where they use an organization structure to define user-role assignment and role-permission assignment [11]. (In the rest of the paper, we use ARBAC to refer to ARBAC97, ARBAC99, and ARBAC02 models). However, these models do not deal with RBAC policies with hybrid hierarchies – where different types of hierarchical relationships among roles co-exist [7]. In a hybrid hierarchy, there are three types of inheritance relationships between any pair roles. The *permission-only inheritance* hierarchy (*I*-hierarchy, $\geq_i$) means that the senior role inherits all permissions of the junior role; The *activation-only inheritance* hierarchy (*A*-hierarchy, $\geq_a$) means that the user who can activate the senior role can also activate the junior role; And the *permission-activation inheritance* hierarchy (*IA*-hierarchy, $\geq$) means both. Several researchers have found that hybrid hierarchy is necessary when a more fine-grained RBAC model is needed. In particular, Joshi *et.al.* show that roles related to *A*-hierarchy can be constrained by a DSoD constraint, and *A*-hierarchy is suitable for permission-centric cardinality constraints, while *I*-hierarchy or *IA*-hierarchy is suitable for user-centric cardinality constraints [10, 13]. Furthermore, Du *et al.* show that hybrid hierarchy is particularly useful when we want to map the policies in multi-domain systems [2].

In this paper, we propose an ARBAC07 model which can deal with more fine-grained security policies using hybrid hierarchies. We achieve this by redefining all the necessary elements in the ARBAC97, ARBAC99, and ARBAC02 models resulting in the ARBAC07$_{97}$, ARBAC07$_{99}$, and ARBAC07$_{02}$ models, respectively. The overall ARBAC07 model is the combination of the three sub-models, as shown in Figure 1. We show that the proposed ARBAC07 model is practical and flexible in complex situations where hybrid hierarchy is needed.

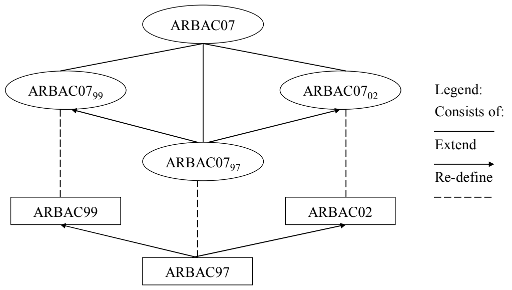


Figure 1. Relationship between ARBAC07 model and ARBAC models

The paper is organized as follows. In section 2, we review the necessary background such as hybrid hierarchy and ARBAC models. We present our ARBAC07 model in Section 3. We present the related work in Section 4 and then conclude in Section 5.

2. Background

2.1. Hybrid Hierarchy

Hybrid hierarchy was introduced in the context of the Generalized Time based RBAC (GTRBAC) model to facilitate specifications of fine grained policies [7]. In a hybrid hierarchy, we have three hierarchy types: *permission-inheritance-only* hierarchy (*I*-hierarchy), *role-activation-only* hierarchy (*A*-hierarchy) and the combined *permission-inheritance-activation* hierarchy (*IA*-hierarchy) [78]. Semantically, $s \geq_i j$ means permissions available through j are also available through s; $s \geq_a j$ means that any user who can activate s can also activate j; and $s \geq j$ means that s inherits permissions of j and the users who can activate s can also activate j. Figure 2 shows a sample hybrid hierarchy.

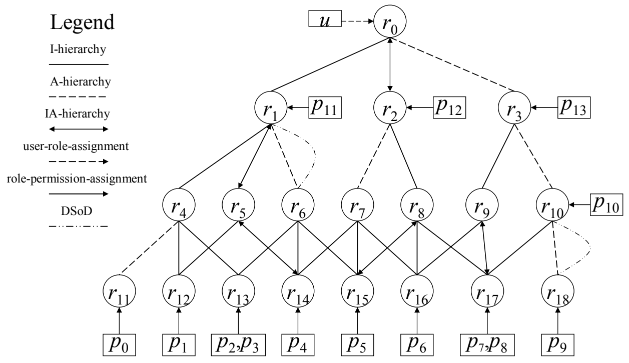


Figure 2. A sample hybrid hierarchy

Joshi *et al.* have shown that in a hybrid hierarchy, the hierarchical relation between any pair of roles which are not directly related could be derived [8]. It is obvious that the three hierarchy types are transitive. For instance, if ($x \geq y$) and ($y \geq z$) then it implies ($x \geq z$). Similarly, since *IA*-relation can be considered as both *I*-relation and *A*-relation, we have the following relations as shown in Figure 3(a): $(x <f_1> y) \wedge (y <f_2> z) \rightarrow (x <f> z)$, *where,* $(<f_1> = \geq) \vee (<f_2> = \geq)$ *and* $<f> = <f_1>$, *if* $<f_2> = \geq$, *otherwise* $<f> = <f_2>$.

A special case of derived relation is when an *A*-relation is followed by an *I*-relation, as shown in Figure 3(b). We should be very careful when analyzing its semantic. Here, by activating x, a user assigned to x can not acquire the permissions of z, although he can acquire the permissions of z by activating y. This means a user assigned to x can still acquire the permissions available through role z even though there is no explicit relation between x and z. In this situation, we say that x has a "*conditioned*" relation with z, written as $x[y] \geq_i z$. In [8], the *conditioned derived* relation is defined as $x[A](B) \geq_i y$, where B indicates a set of A-paths from x to y. In this paper, we ignore set B; if B is not empty, we simply consider it as $x \geq_a y$ without affecting any semantics.

Now consider the case where an *I*-relation is followed by an *A*-relation, as shown in Figure 3(c). Here, a user assigned to x can not acquire the permissions of z, since he can only acquire the permissions of y (by activating x) but can not activate y. Therefore, there's no relation between x and z. We use $\geq_d$ to refer to both regular as well as conditioned derived relations, as defined below:

Definition 2.1 (Derived Relation)*: Let x and y be roles such that* ($x \geq_d y$)*, that is, x has a derived relation with y. Then the following holds:* $(x \geq_i y) \vee (x \geq_a y) \vee (x \geq y) \vee (\exists a \in R, x[a] \geq_i y)$. Here, we say x is senior to y and y is junior to x.

Joshi *et al.* propose a complete and sound set of inference rules to find all the possible derived relations between any pair of roles in a hybrid hierarchy [8].

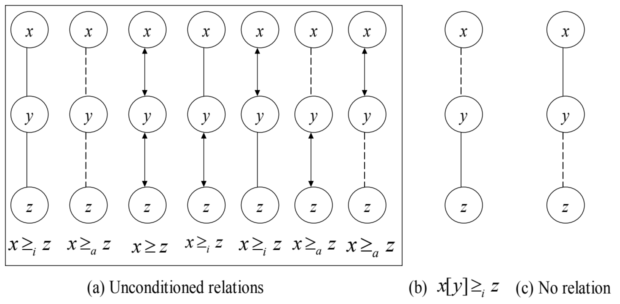
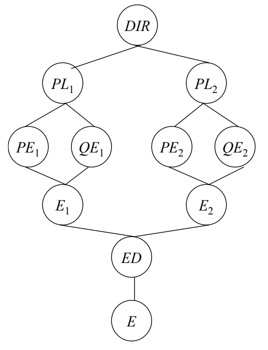


Figure 3. Derived relations between x and z in a hybrid hierarchy

2.2. ARBAC97

We explain the three models of ARBAC97 (URA97, PRA97, and RRA97) by an example using the regular hierarchy and administrative role hierarchy shown in Figure 4 and Figure 5 (both use the standard hierarchy).

URA97 Model: URA97 has two components, one dealing with the assignment of users to roles (the grant model) and the other with revocation of user membership (the revocation model). User-role assignment is controlled in URA97 by the *can_assign* relation: *can_assign*(x, y, z), where x is the administrative role, y is the prerequisite condition, and z is the role range. For example, *can_assign*(PSO_1, ED, [E_1]) means that a member of the administrative role PSO_1 or its senior can assign a user who has current membership in ED to be a member of the regular role E_1. User revocation in URA97 is controlled by the similar *can_revoke* relation: *can_revoke*(x, z), where x is the administrative role, and z is the role range. For example, *can_revoke*(PSO_1, [E_1, PL_1]) means that a member of the administrative role PSO_1 (or a member of an administrative role senior to PSO_1) can revoke a user whose current membership is E_1, PE_1, QE_1, or PL_1



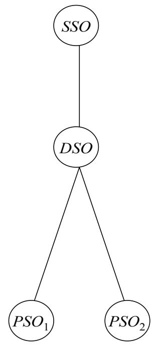


Figure 4. Regular roles **Figure 5.** Administration roles

PRA97 Model: Similar to URA97, PRA97 uses two relations to control the role-permission assignment/revocation: *can_assignp*(x, y, z), and *can_revokep*(x, z), where x is the administration role, y is the prerequisite condition, and z is the role range. For example, *can_assignp*(DSO, DIR, [PL_1,PL_1]) means that a member of the administrative role DSO or its senior can take any permission assigned to the DIR role and make it available to the regular role PL_1.

RRA97 model: RRA97 distinguishes three kinds of roles, as follows:

Abilities:	roles that can only have permissions and other abilities as members.
Groups:	roles that can only have users and other groups as members.
UP-Roles:	roles that have no restriction on membership, i.e., their membership can include users, permissions, groups, abilities, and other UP-roles.

Ability is a collection of permissions that should be assigned as a single unit to a role. For example, the ability to open an account in a banking application will encompass many different individual permissions. It does not make sense to assign only some of these permissions to a role as the entire set are needed to do the task properly. The function of ability is to collect permissions together so that administrators can treat them as a single unit. Assigning abilities to roles is therefore very much like assigning permissions to roles.

Similarly, a *group* is a set of users that should be assigned as a single unit to a role. The function of a *group* is to collect users together so that administrators can treat them as a single unit. Assigning *groups* to roles is therefore very much like assigning users to roles.

In RRA97, operations on UP-Roles are determined by the *can-modify* relation: *can_modify*(x, y), where x is the administrative role and y is the role range. For example, *can_modify*(DSO_1, [E_1, PL_1]) means a member of DSO_1 or its senior can create and delete roles in the range [E_1, PL_1] and can modify relationships between roles in [E_1, PL_1]. RRA97 restricts that ranges in all *can_modify* relations do not overlap. Furthermore, the ranges in all *can_modify* relations must be encapsulated, which means the range has a single senior-most role and a single junior-most role.

2.3. ARBAC99

The RRA97 is unchanged in ARBAC99. ARBAC99 introduces the notion of mobile user/permission and immobile user/permission. An immobile user/permission can be assigned to roles only one time while a mobile user/permission can be assigned to roles several times, as in ARBAC97. For example, if we want to assign *Adam* to *ED* role so that he can be familiar with the basic work in the engineering department but we do not want him to be assigned to any senior roles such as PL_1, we can make *Adam* as an immobile user. Once *Adam* is eligible for other senior roles, we can make *Adam* a mobile user, so that he can be further assigned to senior roles such as PL_1.

2.4. ARBAC02

Again, the RRA97 is unchanged in ARBAC02. ARBAC02 uses two separate role hierarchies called Organization Structure of Users (*OS-U*) and Organization Structure of Permissions (*OS-P*) to control the user-role and role-permission assignments. Unlike the *can_assign*/*can_revoke* operations in URA97, its prerequisite condition x is as defined below:

DEFINITION 2.2 (prerequisite condition for users using OS-U): *a user u is said to satisfy the prerequisite condition x iff*
Case 1: $x \in role$: $\exists (x' \geq x), (u, x') \in URA$
Case 2: $x \in$ *org. unit of OS-U*: $\exists (x' \leq x), (u, x') \in UUA$

Similarly, the prerequisite condition x for a permission is re-defined as follows:

DEFINITION 2.3 (prerequisite condition for permissions using OS-P): *a permission p is said to satisfy the prerequisite condition x iff*
Case 1: $x \in role$: $\exists (x' \leq x)(p, x') \in PRA$

Case 2: $x \in$ *org. unit of OS-P:* $\exists (x' \geq x)(p, x') \in PPA$.

(Note. URA: user-role assignment, UUA: user-organization assignment on OS-U, PRA: permission–role assignment, PPA: permission-organization assignment on OS-P. To distinguish role and organization unit names, we use an '@' in the head of organization unit names.)

Based on these definitions, the *can-assign*, *can-revoke*, *can_assignp*, *can_revokep* remain unchanged.

3. ARBAC07 model

In this section, we propose the ARBAC07$_{97}$, ARBAC07$_{99}$, and ARBAC07$_{02}$ models in detail. These models redefine all the necessary elements in the corresponding ARBAC models to facilitate an RBAC system with hybrid hierarchy.

3.1. ARBAC07$_{97}$

Like the original ARBAC97 model, we still separate the user-role assignment model (URA07$_{97}$), the role-permission assignment model (PRA07$_{97}$), and the role-role administration model (RRA07$_{97}$)

URA07$_{97}$ model: In URA97, *can_assign*(x, y, z) means a member of the administrative role x or its senior can assign a user that satisfies the prerequisite condition y to a regular role in the range z. Because URA07$_{97}$ needs to deal with hybrid hierarchy, we need to redefine the semantics of *can_assign* accordingly. Specifically, we need to characterize the situation in which a user will satisfy a prerequisite condition. Consider the example in Figure 4, any user assigned to *DIR* or PL_1 satisfies the prerequisite condition PL_1 because a member of *DIR* is also a member of PL_1. Therefore, the authors of ARBAC97 implicitly assume that the hierarchy relation in the standard hierarchy is "*Is-a*" relation [10]. In the hybrid hierarchy, it is obvious that "*Is-a*" relation is essentially the *IA*-relation. According to this observation, we re-define the prerequisite condition for users as follows:

Definition 3.1 (Prerequisite condition for users in hybrid hierarchy): *A user u is said to satisfy r iff* $\exists r_1 \geq r$*, such that* $(u, r_1) \in URA$*, where* $\geq$ *is the IA-relation.*

Moreover, we need to specify the notion of range in the hybrid hierarchy. Note that the range is just a convenient representation of a set of roles [14], and it does not have special semantics related to hierarchical relations. We keep the original definition of range in ARBAC07, which is simply a set of roles.

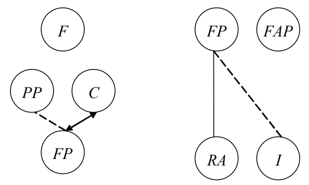
Therefore, in URA07$_{97}$, *can_assign*(x, y, z) means a member of the administrative role x or its "senior" can assign a user that satisfies the prerequiste condtion y to be a member of a regular role in the range z, where the semantics of "senior" and "satisfies" is defined definition 2.1 and 3.1, respectively. We use an example shown in Figure 6(a) to illustrate the semantics of URA07$_{97}$. Figure 6(a) shows the sub-structure of a department in the university. The role *department chair* (*C*) is *IA*-senior to the role *full-time professor* (*FP*) assuming that the *department chair* is also a *full-time professor*. The role *part-time professor* (*PT*) is *A*-senior to *FP* because *PT* should not inherit all permissions of *FP*, but a user assigned to it should be able to activate *FP* when needed. Suppose the university administrator a wants to assign a user u to a *fellowship* role (*F*) (indicating u has been awarded the *fellowship*); furthermore, we add a constraint saying that only the member of the *full-time professor* can be awarded that *fellowship*. This semantics is captured by *can_assign*(a, *FP*, *F*) in URA07$_{97}$. Obviously, any member of *FP* can satisfy the *FP* constraint and can be assigned to *F*. Besides, according to definition 3.1, the member of *C* satisfies the *FP* constraint but the member of *PP* does not satisfy the *FP* constraint; Thus, the member of PP can not be assigned to *F*. This semantics is straight forward in the real organization; the *department chair* is also a *full-time professor* so he can be awarded that *fellowship*; the *part-time professor*, however, is not a *full-time professor* so he can not be awarded that *fellowship*. This example shows that the URA07$_{97}$ model is practical when hybrid hierarchy is needed.

PRA07$_{97}$ Model: PRA07$_{97}$ is the counterpart of URA07$_{97}$. The only difference is that in PRA07$_{97}$, we need to define prerequisite condition required for a permission in presence of a hybrid hierarchy. Consider Figure 4 again; any permission assigned to *E* or *ED* satisfies the prerequisite condition *ED*, since permissions which can be acquired through *E* can also be acquired through *ED*. Therefore, the author of ARBAC97 implicitly assumes that the hierarchy relation in the standard hierarchy is "*Permission Inheritance*" relation [10], which is in conflict with the previous assumption of the "*Is-a*" relation. In the hybrid hierarchy, we can use *I*-relation to accurately capture this semantics. According to this observation, we re-define the prerequisite condition for permissions as follows:

Definition 3.2 (Prerequisite condition for permissions in hybrid hierarchy): *A permission p is said to satisfy r iff* $\exists r_1 \leq_i r$*, such that* $(p, r_1) \in PRA$*, where* $\geq_i$ *is the I-relation.*

Therefore, in PRA07$_{97}$, *can_assignp*(x, y, z) means a member of the administrative role x or its "senior" can assign any permission that "satisfies" the prerequiste condition y to the regular role in the range z, where the semantics of "senior" and "satisfies" is defined by definition 2.1 and 3.2, respectively. We use an example shown in Figure 6(b) to illustrate the semantics of PRA07$_{97}$. Figure 6(b) shows the sub-structure of a department in the university. The role *full-time professor*

(FP) is I-senior to the role *research assistant* (RA) because a *full-time professor* should inherit all permissions of a RA but he need not be able to activate the RA role himself. The role FP is also A-senior to the role *instructor* (I) because FP should not be able to inherit all permissions of I (e.g. a full time professor that is not an instructor can not grade students' exam), but need to activate I when needed. (e.g, when he is an instructor of the course). Now, suppose the university administrator a creates a new role *full-time assistant professor* (FAP) and wants to assign any permission of FP to FAP. This semantics is captured by *can_assignp*(a, FP, FAP) in PRA07$_{97}$. Obviously, any permissions of FP can satisfy the FP constraint and can be assigned to FAP. Besides, according to definition 3.2, the permissions of RA also satisfy the FP constraint but the permissions of I do not satisfy the FP constraint; thus, permissions of I can not be assigned to F. This semantics is staight forward in the real organization. The permissions of RA is also contained in the permissions of FP, and can be assigned to FAP. The permissions of I, however, are not necessarily contained in the permissions of FP, and can not be assigned to FAP. This example shows that the PRA07$_{97}$ model is also practical when hybrid hierarchy is needed.



(a) Example of URA07 $_{97}$ (b) Example of PRA07 $_{97}$

Figure 6. Examples of URA07 and PRA07

RRA07$_{97}$ Model: The key notion in RRA97$_{97}$ is the encapsulated range, which is defined as follows:

DEFINITION 3.3 (Encapsulated range in RRA97$_{97}$): *A range* (x, y) *is said to be encapsulated if* $\forall r_1 \in (x, y) \wedge \forall r_2 \notin (x, y)$, *we have* $r_2 > r_1 \Leftrightarrow r_2 > y$ *and* $r_2 < r_1 \Leftrightarrow r_2 < x$.

Intuitively, the encapsulated range has a single senior-most role and a single junior-most role so any change made to the encapsulated range will not cause unexpected side effects to roles elsewhere in the hierarchy. In the hybrid hierarchy, the hierarchical relation is defined by derived relations $\geq_d$ between any role pair, so we re-define the encapsulated range in hybrid hierarchy as follows:

DEFINITION 3.4 (encapsulated range in RRA07$_{97}$): *A range* (x, y) *is said to be encapsulated if* $\forall r_1 \in (x, y) \wedge \forall r_2 \notin (x, y)$, *we have* $r_2 >_d r_1 \Leftrightarrow r_2 >_d y$ *and* $r_2 <_d r_1 \Leftrightarrow r_2 <_d x$.

In RRA07$_{97}$, we restrict the range in the *can_modify* tuple to be encapsulated range defined in definition 3.4. The operations in RRA07$_{97}$ are the same as in RRA97$_{97}$.

3.2 ARBAC07$_{99}$

ARBAC99 only extends the URA97 and PRA97 by adding the notions of mobile/immobile users/permissions as described in Section 2.3. Obviously, we can still use these notions in ARBAC07$_{99}$ since they have no relationship with hybrid hierarchy.

3.3 ARBAC07$_{02}$

In ARBAC07$_{02}$, we want to adopt the notion of Organization Structure from URA02. Here, we need to characterize the situation under which a user will satisfy a prerequisite condition using organizational structure. As discussed before, the user-role assignment is determined by the IA-relation in the hybrid hierarchy. So we have:

DEFENITION 3.5 (prerequisite condition for users in ARBAC07$_{02}$ using OS-U): *a user u is said to satisfy the prerequisite condition x iff*
Case 1: $x \in role$: $\exists (x' \geq x)\ (u, x') \in URA$
Case 2: $x \in$ *org. unit of OS-U:* $\exists (x' \leq x)\ (u, x') \in UUA$
where $\geq$ *is the IA-relation in hybrid hierarchy.*

Based on these definitions, we can define the same *can_assign* and *can_revoke* operations as in URA02.

Similarly, we adopt the Organization Structure as in PRA02 by capturing the scenario under which a permission will satisfy a prerequisite condition using organization structure. As discussed before, the permission-role assignment is determined by the I-relation in the hybrid hierarchy. Therefore, we have:

DEFENITION 3.6 (Prerequisite condition for permissions in ARBAC07$_{02}$ using OS-P): *a permission p is said to satisfy the prerequisite condition x iff*
Case 1: $x \in role$: $\exists (x' \leq_i x)(p, x') \in PRA$
Case 2: $x \in$ *org. unit of OS-P*: $\exists (x' \geq_i x)(p, x') \in PPA$.
where $\geq_i$ *is the I-relation in hybrid hierarchy*

Given these definitions, we can use the same *can_assignp* and *can_revokep* relations as in PRA97/02.

3.4 Advantages of ARBAC07

The obvious advantage of ARBAC07 is the ability to deal with hybrid hierarchy. As discussed in Section 1, we believe it is very important to design the administration model that can deal with the hybrid hierarchy, which is the main motivation of ARBAC07.

Besides, by using a clearly defined hybrid hierarchy, we are able to solve an ambiguity in the original ARBAC model. Specifically, using standard hierarchy only, the author of ARBAC assumes the relation of standard hierarchy as "Is-a" relation in URA97 and as "Permission Inheritance" relation in PRA97. In ARBAC07, we clearly show that the user-role assignment should be determined by the *IA*-relation and the role-permission assignment should be determined by the *I*-relation in the hybrid hierarchy. Therefore, we resolve this ambiguity in the original model by using hybrid hierarchy.

Finally, we keep all the notions and operations in the original ARBAC models, and redefine them, as needed to provide the semantics for ARBAC07. Hence, our ARBAC07 model keeps all the advantages of the original models, and extends the capability to handle the administration of RBAC models with hybrid hierarchy.

4. Related Work

Several researchers have studied the use of role itself to manage RBAC policies, resulting in several role-based administration models. Sandhu *et al.* propose an ARBAC97 model consisting of URA97, PRA97 and RRA97 [14]. URA97 defines *can-assign* and *can-revoke* relations to manage the user-role assignment, and PRA97 is a counterpart of URA97. The fundamental idea in RRA97 is the encapsulated range, which is a "closed" sub-hierarchy in which every path upwards (or downwards) goes through the same role. They later extend ARBAC97 to ARBAC99, where the notion of mobile and immobile user/permission is introduced. They finally extend ARBAC97 to ARBAC02, where they use the notion of organization structure to redefine the user-role assignment and the role-permission assignment. Crampton *et al.*'s SARBAC model was motivated by the shortcomings of the ARBAC97 model and used the notion of administrative scope [1]. Both of these models use standard hierarchy.

Several researchers have found the limitations of the standard hierarchy. Li *et al.* [9], after careful analysis, have suggested that the standard hierarchy in the RBAC standard introduces several ambiguities. Sandhu first emphasized the necessity of using two different hierarchy relations to allow expressing generic lattice-based policies using RBAC [13]. Moffett *et al.* further show that there are three different uses (semantics) of a role hierarchy [10]. Joshi *et al.* have proposed formal definitions of three types of hierarchies used in this paper [8].

To the best of our knowledge, little research has addressed the issue of how to build a complete administration model for an RBAC system with hybrid hierarchy, which is the primary goal of our paper.

5. Conclusion and Future Work

In this paper, we have proposed the ARBAC07 model that can be used to administer an RBAC system with hybrid hierarchies. Our model re-defines all the necessary elements in the ARBAC model accordingly. Moreover, we find that the original ARBAC model has ambiguous semantics in its user-role assignment and role-permission assignment components, which we remove in our proposed model. We show that our model keeps all the advantages of the original ARBAC model and can deal with more complex situations where hybrid hierarchy is needed. We plan to extend this work to construct a complete administration model for GTRBAC systems with hybrid hierarchy and constraints.

Acknowledgements: This research has been supported by the US National Science Foundation award IIS-0545912.

6. References

[1] J. Crampton, G. Loizou, "Administrative scope: A foundation for role-based administrative models", *ACM Transactions on Information and System Security (TISSEC)*, Volume 6, Issue 2, May. 2003, pp. 201-231.

[2] S. Du, and J. B. D. Joshi, "Supporting Authorization Query and Inter-domain Role Mapping in Presence of Hybrid Role Hierarchy," The 11th ACM Symposium on Access Control Models and Technologies, USA, June 2006.

[3] D. Ferraioio, J. Cugini, and R. Kuhn, "Role-based access control (RBAC): Features and motivations", In Proceedings of 11th Annual Computer Security Application Conference, New Orleans, LA, Dec. 1995, pp. 241-48.

[4] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli, " Proposed NIST standard for role-based access control," *ACM Transactions on Information and Systems Security*, vol. 4, no. 3, pp. 224–274, August 2001.

[5] L. Guiri, "Role-based access control: A natural approach", In Proceedings of the 1st ACM Workshop on Role-Based Access Control, ACM, 1997.

[6] J. B. D. Joshi, E. Bertino, and A. Ghafoor, "Temporal hierarchies and inheritance semantics for GTRBAC", In Proceedings of the 7th ACM symposium on Access control models and technologies, New York, NY, USA, 74–83.

[7] J. B. D. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "Generalized Temporal Role Based Access Control Model," *IEEE Transactions on Knowledge and Data Engineering*, Volume 7, Issue 1, Jan. 2005.

[8] J. B. D. Joshi, E. Bertino, and A. Ghafoor, "Formal Foundations for Hybrid Role Hierarchy", *ACM Transaction in Information and Systems Security* (in press).

[9] N. Li, "A Critique of the ANSI Standard on Role Based Access Control", to appear in *IEEE Security and Privacy*.

[10] J. D. Moffett and E. C. Lupu, "The uses of role hierarchies in access control", Proceedings of the fourth ACM workshop on Role-based access control, 1999, pp. 153-160.

[11] S. Oh, R. Sandhu, "A model for role administration using organization structure", Proceedings of the 7th ACM symposium on Access control models and technologies, Monterey, CA 2002.

[12] R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models", *IEEE Computer* 29(2): 38-47, IEEE Press, 1996.

[13] R. Sandhu, "Role activation hierarchies", Proceedings of the third ACM workshop on Role-based access control, Fairfax, Virginia, United States, 1998, pp. 33-40.

[14] R. Sandhu, V. Bhamidipati, and Q. Munawer, "The ARBAC97 Model for Role-Based Administration of Roles", *ACM Transactions on Information and System Security (TISSEC)*, Volume 2, Issue 1, Feb. 1999, pp. 105-135.

[15] R. Sandhu and Q. Munawer, "The ARBAC99 Model for Administration of Roles (1999)", In Proceedings of 15th Computer Security Applications Conference, Arizona, US, Feb 1999, pp. 229.