ROBAC: SCALABLE ROLE AND ORGANIZATION BASED ACCESS CONTROL MODELS

Zhixiong Zhang
The College Board
Reston, Virginia, USA
jzhang@collegeboard.org

Xinwen Zhang
George Mason University
Fairfax, Virginia, USA
xzhang6@gmu.edu

Ravi Sandhu
George Mason University
and TriCipher Inc., USA
Fairfax, Virginia, USA
sandhu@gmu.edu

***Abstract*—In RBAC, roles are typically created based on job functions inside an organization. Traditional RBAC does not scale up well for modeling security policies spanning multiple organizations. To solve this problem, a family of extended RBAC models called Role and Organization Based Access Control (ROBAC) models is proposed and formalized in this paper. Two examples are used to motivate and demonstrate the usefulness of ROBAC. Comparison between ROBAC and other related RBAC models is given. We show that ROBAC can significantly reduce administration complexity for Web and Internet-based applications involving a large number of organizations. Some administrative issues for ROBAC are identified and discussed. Although the theoretical-expressive power of ROBAC is the same as that of RBAC, it is more succinct and intuitive to use ROBAC than to use RBAC when applications involve many organizations.**

Key words: *access control, RBAC, role and organization based access control, ROBAC.*

I. INTRODUCTION

The adoption of RBAC in commercial software and enterprises has rapidly increased in recent years [7]. The complexity of an RBAC system can be defined on basis of the number of roles, the number of permissions, the size of the role hierarchy, the constraints on user-role and permission-role assignments, etc. [10]. For existing large-scale RBAC systems, the number of roles and the number of permissions are in the order of 1000s. Beyond that magnitude, the performance of RBAC may degrade and its management becomes too difficult to handle correctly.

Several approaches have been proposed to scale up RBAC systems. Giuri and Iglio [3] extend RBAC by introducing the concept of role templates with parameterized privileges to achieve content-based access control. Thomas [14] proposes Team Based Access Control (TMAC) to scale up permission assignment with fine-grained run-time permission activation at the level of individual users and objects. Perwaiz and Sommerville [6] describe a mechanism for viewing role-permission relationships in the context of organizational structures, which reduce the number of roles in an RBAC implementation. Sandhu and Park [12] apply the URA97 model in ABAC97 [8] to intranet web applications to achieve decentralized administration of user-role assignment. Park et al. [5] proposed a composite RBAC for large and complex organizations.

Much of this previous work addresses RBAC in the context of a single organization and is mainly motivated by B2E (Business to Employee) applications. On the other hand, B2B (Business to Business) and B2C (Business to Consumer) applications often involve a large number of organizations such as corporations, schools, families, etc. Typically, users from different organizations with same role name have slightly different access privileges due to local variations in policy details. Using traditional RBAC naively in these situations can result in an enormous number of roles and permissions, well into the order of millions.

To address the RBAC scalability problem in such situations, a new family of models called Role and Organization Based Access Control (ROBAC) models is proposed and formalized in this paper. We emphasize that this paper is focused on the ROBAC model. Discussion of implementation issues is beyond the scope of this paper and will be covered in future publications. More specifically in the terminology of the recently introduced PEI framework [13], this paper focuses on the policy model layer while enforcement and implementation models are left for future publications.

The rest of this paper is structured as follow. Section 2 gives two motivating examples for ROBAC models. Formal definitions of ROBAC models are presented in Section 3. Some administrative issues are discussed in Section 4. Related

1-4244-0429-0/06/$20.00 ©2006 IEEE

work and a detail comparison between RBAC and ROBAC are given in Section 5. Section 6 summarizes this paper and presents future research directions.

II. MOTIVATING EXAMPLES

We begin with two motivating examples from B2B and B2C context respectively. These are abstracted from the authors experience with similar real-world applications.

B2B example: This example considers access control policy for a web based report delivery system, which only allows authorized users to access specific reports. The users are educational professionals from schools, districts, and states in USA[1]. There are on the order of 10,000 schools participating in the system. Reports are classified into types based on sensitivity and nature of the content. Because some report types include privacy-sensitive data such as student test results and personal information, only an authorized user, say, School_1's official, can view School_1's reports but cannot view School_2's reports. There are many different types of reports, each of which may have up to three different levels of information (state level, district level, and school level). Some sample report types are listed in Table I.

TABLE I. SAMPLE REPORT TYPES IN B2B EXAMPLE

Type A Report (school level, district level, and state level)
Type B Report (school level only)
Type C Report (school level only)
Type D Report (school level only)
Type E Report (school level and district level)
Type F Report (district level and state level)
…

States, districts, and schools usually form a management hierarchy. Fig. 1 shows an example of a possible management hierarchy among states, districts, and schools.

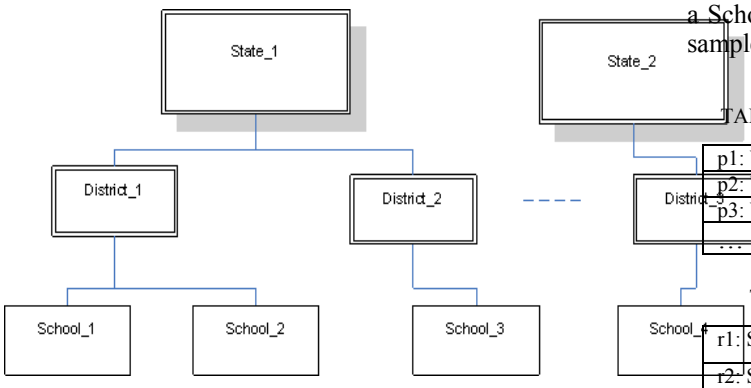

Figure 1. A sample organization hierarchy

The informal description of some security policies of the system may include:

- Users from a school are only allowed to access the reports related to this school.
- Users from a district education office are allowed to access the reports related to this district and the schools under it.
- Users from a state education office are allowed to access the reports related to this state and the districts and schools under it.
- School principles can view type A and type B reports.
- School teachers can view type B and type E reports.
- Officials from a district's board of education offices can view type A and type B reports but cannot view type D reports

Under the above policies, an authorized school level user (say School_1 teacher) can only access certain types of the user's own school's reports, but is not allowed to access other types of reports, and, further, cannot access other school's or any district or state level reports[2]. An authorized district level user can access certain types of the user's own district's reports (district level) and may also access the same types of its subordinate schools' reports. For example, an authorized District_1 official can access District_1's district level Type_A reports and school level Type_A report for School_1 and School_2 since the School_1 and School_2 are under District_1.

Assuming there are 10,000 organizations and 10 types of reports, if we use traditional RBAC (RBAC96 [9] or NIST-RBAC [10]) to model this problem directly, we have to define about 100,000 (10,000 x 10) permissions because viewing School_1's Type_A report is different from viewing School_2's Type_A report. We also need to define 100,000 different roles because a role that can view a School_1_Type_A report is different from a role that can view a School_2_Type_A report. Table II and Table III show some samples of the possible permissions and roles in this example.

TABLE II. SAMPLE PERMISSIONS IN B2B EXAMPLE(WITH RBAC)

p1: View School_1 Type A Report
p2: View School_2 Type A Report
p3: View District_1 Type A Report
…

TABLE III. SAMPLE ROLES IN B2B EXAMPLE(WITH RBAC)

r1: School_1 Type A Report Viewer with permission p1.
r2: School_2 Type A Report Viewer with permission p2.
r3: District_1 Type A Report Viewer with permission p3.
…

[1] The stipulation of USA allows us to be more concrete and realistic about this example.

[2] We are assuming that access not explicitly allowed by the stated policies is denied.

Our goal here is not so much to define a complete and coherent policy for this example but rather to illustrate the issues and complexities involved.

B2C example: Consider an online subscription-based tutoring system, where customers are families that have children in elementary schools. Parents pay subscription fees for their children and are authorized to create/update the family's profile and view their children's progress reports. Students that have subscribed to the service can take classes on the web and view their progress reports and family profiles. Here, family profiles and student's progress reports need to be protected against unauthorized access. There are potentially millions of families, and even 10s or 100s of millions.

The informal description of some security policies of this system may include:

- Parents can only view their own children's progress reports.
- Parents can create/update/view their family's profile.
- A student can view his/her own progress report and view his/her family's profile.

Suppose we use traditional RBAC to support these policies. Because Family_1's parent is only allowed to access Family_1's profile and Family_1's children's progress reports, the Family_1's parents have slightly different permissions from that of Family_2's parents. Table IV and Table V show some samples of the possible permissions and roles when using traditional RBAC in this B2C example.

TABLE IV. SAMPLE PERMISSIONS IN B2C EXAMPLE(WITH RBAC)

p1: Update Family_1's Profile
p2: View Family_1's Kids' Progress Reports
p3: View Family_1's Profile
p4: Update Family_2's Profile
p5: View Family_2's Kids' Progress Reports
p6: View Family_2's Profile
……

TABLE V. SAMPLE ROLES IN B2C EXAMPLE(WITH RBAC)

r1: Family_1 Parents permission p1 and p2.
r2: Family_1 Student permission p2 and p3.
r3: Family_2 Parents permission p4 and p5.
r4: Family_2 Student with permission p5 and p6.
…

We can see that the administrative complexity is very high in applying RBAC directly to the above two examples. These scenarios are quite typical for B2B and B2C applications. In practice, security and application engineers usually work around this problem by combining RBAC with other access control mechanisms such as context-based or attribute-based access control. The result is an ad hoc access control model with a specialized administrative tool for each application [4, 11].

III. ROBAC MODELS

To address the issue that traditional RBAC [9, 10] does not scale up well for applications involving multiple organizations where access privileges are based on both roles and organizations, we extend RBAC to ROBAC (Role and Organization Based Access Control) by basing access decision on both role and organization.

The central idea behind ROBAC is quite simple. Instead of only using role related information, ROBAC utilizes both the role information and the organization information during the authorization process. Specifically, in ROBAC a user is assigned to role and organization pairs instead of roles only. Moreover, the permissions in ROBAC are defined as operations over object types instead of operations over objects only. A user can access an object if and only if the user is assigned to a role and organization pair, and the role has the right to access the object's type and the organization "owns" the object. In the following sections, we show that the number of roles and permissions for the above B2B and B2C examples can be reduced significantly if we use ROBAC to model them. This demonstrates that ROBAC can reduce administrative complexity significantly for applications involving a large number of similar organizations.

For easy comparison with traditional RBAC (RBAC96) [9], we define ROBAC models one by one based on the increasing security functionality of the models ($ROBAC_0$, $ROBAC_1$, $ROBAC_2$, $ROBAC_3$) in direct correspondence with the four models of well-known RBAC96 family ($RBAC_0$, $RBAC_1$, $RBAC_2$, $RBAC_3$). $ROBAC_0$ is a base model. $ROBAC_1$ is $ROBAC_0$ plus role hierarchy and organization hierarchy. $ROBAC_2$ is $ROBAC_0$ plus constraints. $ROBAC_3$ is $ROBAC_0$ plus role hierarchy, organization hierarchy and constraints. Fig. 2 shows the relationship of the ROBAC models and Fig. 3 portrays their essential characteristics.

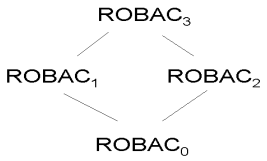

Figure 2. Relationship among ROBAC models

To make the paper concise, we only give the formal definitions for $ROBAC_0$ and $ROBAC_1$ here. Definitions for $ROBAC_2$ and $ROBAC_3$ can be similarly developed.

Definition 1: $ROBAC_0$ has the following components:

- U -- a set of users (same as U in RBAC96);
- S -- a set of sessions (same as S in RBAC96);
- R -- a set of roles (same as R in RBAC96);
- O -- a set of organizations;

- Op -- a set of operations;
- A -- a set of assets;
- At -- a set of asset types;
- $P \subseteq Op \times At$ -- a set of permissions;
- $RO \subseteq R \times O$ -- a set of applicable role and organization associations;
- $PA \subseteq P \times R$ -- a many-to-many permission-to-role assignment relation;
- $UA \subseteq U \times RO$ -- a many-to-many user-to-role-and-organization assignment relation;
- *user*: $S \rightarrow U$ -- a function mapping a session s_i to a single user *user*(s_i) (same as *user* in RBAC96);
- *atype*: $A \rightarrow At$ -- a function mapping an asset to its type;
- *aorg*: $A \rightarrow O$ -- a function mapping an asset to the organization it belongs to;
- *assigned_role-orgs*: $U \rightarrow 2^{RO}$ -- a function mapping a user to a set of role-organization pairs assigned to the user; *assigned_role-orgs(u)* = $\{ (r,o) \mid (u, (r,o)) \in UA \}$;
- *active_role-orgs*: $S \rightarrow 2^{RO}$ -- a function mapping a session s_i to a set of active role-organization pairs; *active_role-orgs*(s_i) $\subseteq$ *assigned_role-orgs*(*user*(s_i));
- *can_access*(S, Op, A) -- a predicate defined as *can_access(s, op, a)* is true iff $\exists (r, o) \in$ *active_role-orgs(s)* $\wedge$ *aorg(a)* = $o \wedge ((op, atype(a)), r) \in PA$;

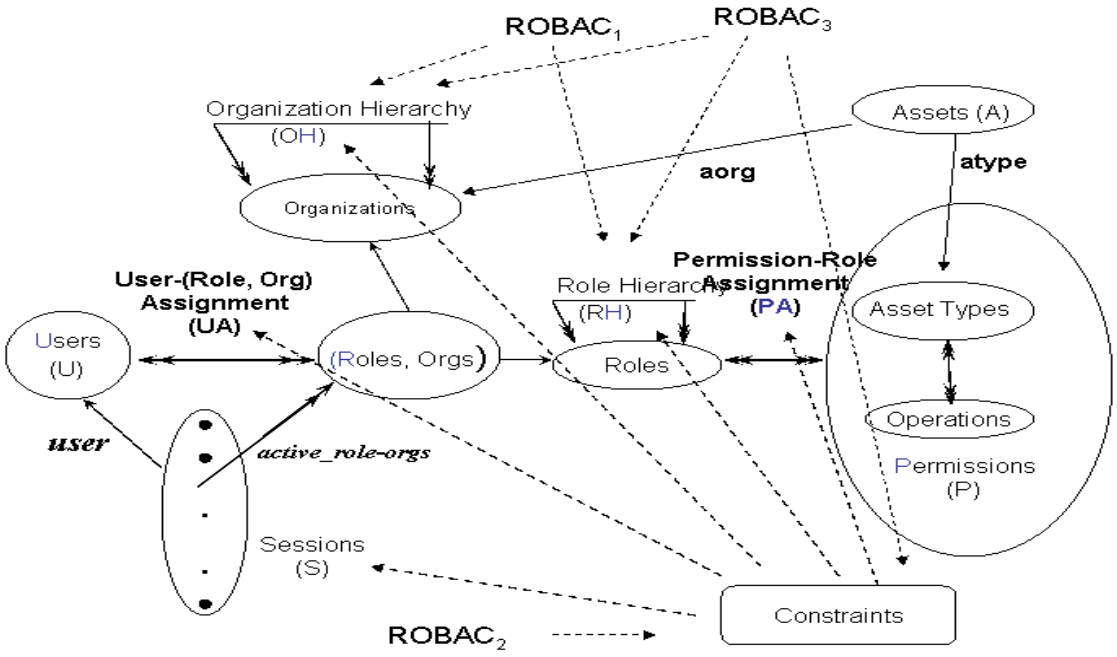

Figure 3. A family of ROBAC models

Where the operations (Op) are similar to the operations or actions in some traditional RBAC [2]; the assets (A) are similar to objects; the active_role-orgs is used to model activation of role-organization pairs inside a session. Briefly, $ROBAC_0$ extends $RBAC_0$ by:

- introducing new sets: Organizations(O), Asset (A), Asset Types (At), Operations (Op), and Role-Organization pairs (RO);
- introducing new functions: *atype, aorg, assigned_role-orgs, active_role-orgs*;
- redefining permissions (P) and user to role assignment (UA);
- introducing a predicate *can_access(s, op, a)*.

Predicate *can_access(s, op, a)* is true means that the session *s* or user *user(s)* can perform operation *op* on asset *a* during the session. The definition of *can_access* in $ROBAC_0$ indicates that a user *(user(s))* in a session *s* can perform an operation *op* over an asset *a* if and only if that the user has an active role and organization pair *(r, o)* in that session and the *r* has a permission to perform the *op* over the *a*'s type and the *a* is related to the *o*.

Definition 2: $ROBAC_1$ has the following components:

- U, S, R, O, Op, A, At, P, RO, PA, UA, *user, atype, aorg* are same as those from $ROBAC_0$.
- $OH \subseteq O \times O$ -- a partial order relation on O called organization hierarchy;
- $RH \subseteq R \times R$ -- role hierarchy (same as RH in RBAC96);
- *assigned_role-orgs*: $U \rightarrow 2^{RO}$ -- a function mapping a user to a set of role-organization pairs assigned to the user; *assigned_role-orgs(u)* $= \{ (r,o) \mid \exists r' \geq r \wedge \exists o' \geq o \wedge (u, (r',o')) \in UA \}$;
- *active_role-orgs*: $S \rightarrow 2^{RO}$ -- a function mapping each session s_i to a set of active role-organization pairs; *active_role-orgs*(s_i) $\subseteq$ *assigned_role-orgs(user*(s_i)*)*;
- *can_access*(S, Op, A) – a predicate defined as *can_access(s, op, a)* is true iff $(r, o) \in$ *active_role-orgs(s)* $\wedge$ *aorg(a)* $\leq o \wedge (\exists r' \leq r, ((op, atype(a)), r') \in PA)$;

$ROBAC_1$ adds OH (organization hierarchy) and RH (role hierarchy) and changes *assigned_role-orgs* function and *can_access* predicate from $ROBAC_0$.[3]

The definition of *can_access* in $ROBAC_1$ means that a user *user(s)* in a session *s* can perform an operation *op* over an asset *a* if and only if that the user has an active role and organization pair *(r, o)* in that session and the role *r* or any of its junior roles has a permission to perform the operation *op* over the asset *a*'s type and the asset *a* belongs to the organization *o* or any of its subordinate organizations.

$ROBAC_2$ is $ROBAC_0$ plus constraints. In ROBAC, we can define constraints on the RO relation in addition to role activations (sessions), UA, and PA. There are two levels of constraints, global constraints and local constraints. We explain these two levels of constraints by using the most common UA constraints: separation of duty.

Separation of duty (SOD) constraint I

A separation of duty constraint I is an irreflexive binary relation on R ($SOD \subseteq R \times R$), which is effective in all organizations. That is, if $(r_i, r_j) \in SOD$, then no user can be assigned to both r_i and r_j for each organization in O.

Separation of duty constraint II

A separation of duty constraint II is an irreflexive binary relation on RO ($SOD \subseteq RO \times RO$), which is only effective in the specified organizations. That is, if $((r_i, o_k), (r_j, o_l)) \in SOD$, then no user can be assigned to r_i in organization o_k and r_j in organization o_l.

[3] In $ROBAC_1$, *assigned_role-orgs* function and *can_access* predicate consider both role hierarchy and organization hierarchy. We could have finer classification of ROBAC models wherein only one of these hierarchies is allowed in two subcases of ROBAC1, one where organization hierarchies are allowed and one where role hierarchies are allowed.

In **Separation of duty constraint I**, constraints are defined on role set R only. We call these kinds of constraints as global constraints since they apply to all organizations. In **Separation of duty constraint II**, constraints are defined on role and organization pair set RO. We call these kinds of constraints as local constraints since they only apply to specified organizations.

The global static SOD in ROBAC means that the same individual user can never hold mutually exclusive roles within the same organization. The local static SOD in ROBAC means that the same individual user can never hold mutually exclusive role-organization pairs. To model static SOD, the set UA and the function *assigned_role-orgs* needs to be modified to reflect the constraints.

The global dynamic SOD in ROBAC means that the same individual user can never hold mutually exclusive roles for same organizations in a session. The local dynamic SOD in ROBAC means that the same individual user can never hold mutually exclusive role-organization pairs in a session. To model the dynamic SOD, the function *active_role-orgs* needs to be modified to reflect the constraints.

A formal definition of $ROBAC_2$ is omitted due to lack of space.

$ROBAC_3$ is the consolidated model of $ROBAC_1$ and $ROBAC_2$. $ROBAC_3$ may have some additional constraints on role hierarchy (RH) and organization hierarchy (OH).

For the aforementioned B2B example, we can use $ROBAC_1$ to model it very conveniently. We show some ROBAC elements differing from those in RBAC as follows.

- O= {State_1, State_2, District_1, District_2, District_3, School_1, School_2, School_3, School_4, …}
- OH = {(State_1, District_1), (State_1, District_2), (District_1, School_1), (District_1, School_2), (District_ 2, School_3), (State_2, District_3), (District_3, School_4), …}
- At = {Type_A_Report, Type_B_Report, …}
- RO = { (r1, District_1), (r2, District_1), (r1, School_1), (r2, School_2), (r3, School_1), (r4, School_1), … }

Possible permissions and roles are listed in Table VI and Table VII.

TABLE VI. SAMPLE PERMISSIONS IN B2B EXAMPLE (WITH ROBAC)

p1: View Type A Report
p2: View Type B Report
p3: View Type C Report
p4: View Type D Report
…

TABLE VII. SAMPLE ROLES IN B2B EXAMPLE (WITH ROBAC)

r1: Type A Report Viewer which has permission p1.
r2: Type B Report Viewer which has permission p2.
r3: Type C Report Viewer which has permission p3.

r4: Type D Report Viewer which has permission p4.
…

Based on the security policies, r1 and r2 can have role-organization pairs with all levels of organizations but r3 and r4 can only have role-organization pairs with school level organizations.

For the aforementioned B2C example, we can use $ROBAC_0$. Possible permissions and roles are listed in Table VIII and Table IX.

TABLE VIII. SAMPLE PERMISSIONS IN B2C EXAMPLE (WITH ROBAC)

p1: Update Family Profile
p2: View Kid's Progress Reports
p3: View Family Profile
…

TABLE IX. SAMPLE ROLES IN B2C EXAMPLE (WITH ROBAC)

r1: Parent which has permission p1 and p2.
r2: Student which has permission p2 and p3.
…

Comparing to RBAC, the number of roles and permissions in RODBC are reduced dramatically in the above B2B and B2C examples. The set of applicable role and organization pairs (RO) is a newly introduced concept in ROBAC. RO is normally constructed implicitly via some predefined rules. The size of RO may be large when there are a large number of organizations involved, but the administrative effort on RO is not big. We will discuss the administrative issues in the next section. This demonstrates that using ROBAC to model the above B2B and B2C examples is more succinct and intuitive than using RBAC.

IV. ADMINISTRATIVE ISSUES IN ROBAC

The access control for administrative tasks in ROBAC can be accomplished by using regular RBAC. The general approaches of ARBAC97 [8] and its variants can be used to administer ROBAC. Here we highlight some difference and new features to demonstrate the benefits of using ROBAC from the administrative point of view.

In a ROBAC system, O and OH are expected to be mostly static and change very slowly. They are typically available in advance and will not change frequently after the ROBAC is set up. So the major administrative effort is on the permission-to-role assignment and user-to-role-and-organization assignment.

A. Permission-to-role Assignment

In traditional RBAC, permissions are normally defined as operations over objects [2]. In ROBAC, permissions are defined as operations over asset (object) types.

In the aforementioned B2B example, ROBAC only creates one role called Type_A_Report_Viewer which has one permission called View_Type_A_Report for viewing type A report, but a traditional RBAC needs to create a role for each organization's type A report viewer, such as School_1_Type_A_Report_Viewer which has View_School_1_Type_A_Report permission, and School_2_Type_A_Report_Viewer which has View_School_2_Type_A_Report permission, etc.

In the B2C example, ROBAC only creates two roles (parent and student) instead of a parent role and a student role for each family in the traditional RBAC. Permissions in ROBAC are defined as a subset of Op x AT while permissions in traditional RBAC are defined as a subset of Op x A. Normally, |AT| is much smaller than |A|. In the above B2B example,

$$|A| \approx 10{,}000 \times |AT|$$

So the number of roles and permissions in ROBAC is much smaller than that in RBAC.

After defining the roles and permissions in ROBAC, PRA97 [8] can be used to perform Permission-to-Role assignment. Under the situations similar to the above two examples, the administrative complexity in Permission-to-Role assignment is significantly reduced because of the reduction of the number of roles and permissions in ROBAC,

B. User-to-role-and-organization Assignment

Sandhu and Park [12] address User-to-Role assignment for intranet web applications by applying the URA97 model. For intranet web applications, we normally know user identifiers in advance but it is not always true for Internet-based web applications. We cannot apply URA97 model when the user identifiers are not available to security administrators in advance. Because there are a very large number of users, we use indirect user to role-organization assignment (IUROA) methods. The basic idea is that assigning a user to role-organization pairs is based on something that the user knows and/or holds. We also utilize some approaches in Rule-based RBAC [1] to perform automatic user-role assignment based on user attributes. The detailed discussion of IUROA belongs to implementation models in the PEI framework and is out of scope of this paper.

As the decision logic (*can_access* predicate) in ROBAC is deterministic, we believe that it is viable to develop a general-purpose authorization engine and an administration tool based on our proposed ROBAC models.

C. Role-Organization Pairs Administration

The size of RO in ROBAC may become large if there are a large number of organizations involved. Instead of creating RO explicitly, we can define RO implicitly by using some rules. For example, in the aforementioned B2B example, we use the following rules to establish RO implicitly:

- r1 (Type A Report Viewer) and r2 (Type B Report Viewer) can associate with any organizations.
- r3 (Type C Report Viewer) and r4 (Type D Report Viewer) can only associate with school type organizations.

For the B2C example, we allow any role associate to any organization. While the size of RO may be large but the administrative work for RO is small.

V. Discussion and Related Work

The organization concept in ROBAC introduces a powerful abstraction that can be coupled quite naturally with the traditional concept of roles. For example, we can treat the divisions or project teams in an enterprise as organizations. So ROBAC can be used in many B2E applications. Because ROBAC performs access control based on both roles and association relations between users and protected resources (assets), it is suitable to model privacy-related policies.

In this section we first compare ROBAC with RBAC96, then with some existing access control models which extend RBAC with similar motivations to ROBAC to some degree.

A. Comparison with RBAC96

Fig. 4 shows the traditional RBAC96 model [9].

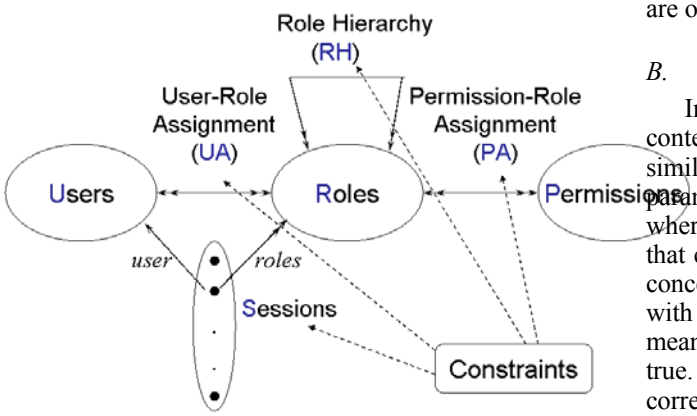

Figure 4. RBAC96 Model [9]

From Fig. 3 and Fig. 4, we can see that permissions in the RBAC96 are very abstract while permissions in ROBAC are more concrete. Users in RBAC96 are assigned to roles while users in ROBAC are assigned to role-organization pairs. Detailed comparison between these two models is listed in Table X.

TABLE X. Comparison between RBAC and ROBAC (with N Organizations and M asset types)

	RBAC	ROBAC
Number of permissions	N×M	M
Number of roles	N×M	M
Organization hierarchy	N/A	Yes
Role hierarchy	Yes	Yes
Constraints	Yes	Yes
User-role-(org) assignment	URA97	URA97+IUROA
Permission-role assignment	PRA97	PRA97
Role administration	RRA97	RRA97
Number of role-org pairs	N/A	≤ N×M
Role-org pairs administration	N/A	Implicitly and rule based

It is obvious that for any RBAC model, we can construct a ROBAC model with only one organization and make all assets belong to the organization and define a one-to-one mapping between assets (A) and asset types (At). The constructed ROBAC has the same security behavior as the original RBAC. So ROBAC inherits all RBAC's benefits such as policy neutrality, principle of least privilege, separation of duty, and easy management.

Similarly, for any ROBAC model, we can construct an equivalent RBAC model. Although these two facts imply the same theoretical-expressive power between ROBAC and RBAC, the practical benefits of using ROBAC over RBAC for applications involving a large number of similar organizations are obvious.

B. Comparison with Role Templates

In the perspective of restricting access to a subset of contents, the role templates [3] proposed by Giuri and Iglio has similar effectiveness as ROBAC. In a role template, a parameterized privilege is defined as (*am, o, exp(v_1, …, v_n)*) where *am* is an access mode (same as an operation in ROBAC) that can be performed on object *o* which is similar to the asset concept in ROBAC. The *exp(v_1, …, v_n)* is a logical expression with unbound variables v_1, …v_n. The parameterized privilege means to perform operation *am* over *o* while *exp(v_1, …, v_n)* is true. The unbound variables in the parameters of the corresponding role template need to be bound when a role is assigned to a user. Although role templates can model security policies in our B2C example by defining organization, objects, and users as unbound variables and "belong to" as a logical expression, the approach is not so straightforward as with ROBAC.

Because the values of the variables in role templates are un-structured, it is difficult to use role templates to model the B2B example aforementioned. In general, role templates can achieve the modeling capability of $ROBAC_0$ with similar complexity, but it is very difficult for role templates to achieve the modeling capability of $ROBAC_1$ due to the flexibility introduced by organization hierarchies used in these ROBAC models. We believe that administrative tasks in ROBAC are much simpler than those in role templates approach.

C. Comparison with Team-based Access Control (TMAC)

In TMAC [14], the notion of "team" is proposed as an abstraction that groups users in specific roles with the objective of accomplishing a specific task or goal. As recognized by the author, it is very difficult for traditional RBAC to enforce the following two requirements at same time: scalable permission

assignment and fine-grained, run-time permission activation at the level of individual users and objects. The notion of "team" in TMAC and the notion of "organization" in ROBAC are different. Team in TMAC or C-TMAC [4] represents a group of users and has roles and permissions derived from the roles and permissions of the users in the group while organization in ROBAC does not have roles or permissions. The teams in TMAC are flat while the organizations in ROBAC can be structured. In TMAC, an access decision is based on a team's permissions, the user's contexts, and the object's contexts, while in ROBAC a decision is based on user's roles and the indirect association between the user and the assets via organizations. Both TMAC and ROBAC realize the importance to distinguish object (asset) type and object instance (asset) for scalable permission assignment in RBAC. With TMAC we find that it is possible to model the B2C example, but it is very complex to model the B2B example due to the presence of organization hierarchy.

Further, we can simulate most components (except for the special role team header) of TMAC with a $ROBAC_0$ model directly. For the team header role in TMAC, we can simply create a role in ROBAC to model it.

D. Comparison with Organizational Units

Perwaiz and Sommerville [6] utilize organizational context to restrict the permissions of roles. An organization unit (OU) in their paper has its own permissions. The maximum permissions of a role in an organization unit are the intersection of the role's permissions and the organization unit's permissions. So the OU in Perwaiz's approach is more like the "team" notion in the TMAC than to the "organization" in ROBAC, since an organization in ROBAC does not associate with permissions. The access control decision process in ROBAC is also different from that in [6]. In the perspective of reducing the number of roles in RBAC, Perwaiz and Sommerville's approach has some similar effect of ROBAC. But the administration of ROBAC is simpler since permissions are not assigned to organizations.

E. Comparison with Organization Entity in Credential Based Access Control

Biskup and Wortmann [15] discussed a credential-based implementation of compound access control policies. In which, they introduce a concept called "organizational entity" which is a collection of objects (assets) those may belong to different owners. Organizational entities contain at least one object and can be both overlapping and nested. For each organization entity, there is a unique controller who performs administrative works on the organization entity. The controller explicitly states grantees or he implicitly states them with the help of assigners (trusted agents). The controller also acts as verifier to regulate actual access requests concerning his organizational entity. The permissions or capability or interface are defined in the form of $<o, a>$ (perform an action a over object o) which is similar to the permission defined in traditional RBAC [2]. The organization entity in [15] acts like a namespace while the organization in ROBAC acts as an indirect association between users and assets (objects). The access decision logic in [15] is also quite different from the access decision logic in ROBAC. Credentials based approaches can be used to implement IUROA (Indirect User to Role and Organization Assignment) in ROBAC by assigning user to role-organization pairs based on the credentials the user holds.

VI. Conclusions and Future Work

A family of extended RBAC models called Role and Organization Based Access Control (ROBAC) models is proposed and formalized in this paper. The motivation behind ROBAC is to scale up RBAC for B2B and B2C applications where a large number of organizations are involved. The advantages of ROBAC models over traditional RBAC models are shown via two examples. Some administrative issues in the context of ROBAC have been discussed. A comparison between RBAC and ROBAC has been given. We show that the benefits of using ROBAC increases in proportion to the number of organizations involved. We claim that ROBAC is more intuitive and succinct than many RBAC variants when used for scenarios involving a large number of organizations.

A partial implementation of a $ROBAC_1$ model has been successfully used as an authorization engine, which provides authorization services for multiple web applications on Internet [16]. The enforcement and implementation aspects of ROBAC will be explored in future work.

References

[1] [1] Mohammad Abdullah Al-Kahtani, "A Family of Models for Rule-Based User-Role Assignment", PhD Dissertation, George Mason University, Spring 2004.

[2] [2] David F. Ferraiolo, John F. Barkley, D. Richard Kuhn, "A role-based access control model and reference implementation within a corporate intranet," ACM Transactions on Information and System Security (TISSEC), Volume 2 Issue 1, February 1999.

[3] [3] Luigi Giuri and Pietro Iglio, "Role Templates for Content-Based Access Control", Proceedings of Second ACM Workshop on Role-Based Access Control, November 1997.

[4] [4] Christos K. Georgiadis, Ioannis Mavridis, George Pangalos, and Roshan K. Thomas, " Flexible Team-Based Access Control Using Contexts", SACMAT'01, May 3-4, 2001, Chantilly, Vriginia, USA.

[5] [5] Joon S. Park, Keith P. Costello, Teresa M. Neven, Josh A. Diosomito, "A Composite RBAC Approach for Large, Complex Organizations", SACMAT'04, June 2-4, 2004, Yorktown Heights, New York, USA.

[6] [6] Najam Perwaiz and Ian Sommerville, "Structured Management of Role-Permission Relationships", SACMAT'01, May 3-4, 2001, Chantilly, Vriginia, USA.

[7] [7] RTI International, "The Economic Impact of Role-Based Access Control", March 2002, http://www.nist.gov/director/prog-ofc/report02-1.pdf

[8] [8] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer, The ARBAC97 Model for Role-Based Administration of Roles, ACM Transactions on Information and Systems Security, Volume 2, Number, February 1999.

[9] [9] Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Youman, Role-Based Access Control Models, IEEE Computer, Volume 29, Number 2, February 1996.

[10] [10] Ravi Sandhu, David Ferraiolo, and Richard Kuhn, The NIST Model for Role-Based Access Control: Towards A Unified Standard, National Institute of Standards and Technology, December 2000, http://csrc.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf

[11] [11] Andreas Schaad, Jonathan Moffett, Jeremy Jacob, "The Role-Based Access Control System of a European Bank: A Case Study and Discussion", SACMAT'01, May 3-4, 2001, Chantilly, Vriginia, USA.

[12] [12] Ravi Sandhu , Joon S. Park, Decentralized user-role assignment for Web-based intranets, Proceedings of the third ACM workshop on Role-based access control, p.1-12, October 22-23, 1998, Fairfax, Virginia, United States

[13] [13] Ravi Sandhu, Kumar Ranganathan, and Xinwen Zhang, “Secure Information Sharing Enabled by Tusted Computing and PEI Models”, ASIACCS ’06, March 2006, Taipei, Taiwan.

[14] [14] R.K. Thomas, “Team-Based Access Control (TMAC): A Primitive for Applying Role-Based Access Controls in Collaborative Environments”, Proceedings of the Second ACM workshop on Role-based Access Control, Fairfax, VA, USA, 1997.

[15] [15] Joachim Biskup and Sandra Wortmann, "Towards a credential-based implementation of compound access control policies", Proceedings of the ninth ACM symposium on Access control models and technologies, June 2004

[16] [16] The College Board web site for professional customers, https://epl.collegeboard.com/epl/goHome.do