

## An Organizational Structure-Based Administration Model for Decentralized Access Control\*

SEJONG OH, CHANGWOO BYUN<sup>+</sup> AND SEOG PARK<sup>+</sup>

*Department of Computer Science*

*Dankook University*

*Cheonan, 330-714 South Korea*

<sup>+</sup>*Department of Computer Science*

*Sogang University*

*Seoul, 121-742 South Korea*

We propose an effective administration model using organizational structure for a decentralized role-based access control environment. Access control administration is a critical issue for large organizations and information systems. A large organization needs decentralized access control by multiple security officers because it has many users and information objects, and a single security officer cannot do all the work. If an organization has multiple security officers, managing them is another important security task. The task includes defining the authority scope and keeping the administrative operations of each security officer legal. Access control administration means controlling security officers' administrative work. ARBAC is a typical model for access control administration. ARBAC defines authority scope using the role hierarchy, and it leads many shortcomings. Our proposed model uses the organizational structure as a basis for defining authority scope and keeping administrative operations legal. The proposed model overcomes the shortcomings of ARBAC, and offers a clear rationale for access control administration.

**Keywords:** access control, role, organization, organizational structure, security

### 1. INTRODUCTION

Since 1970, as many companies and public organizations have recognized that the computer is an important aspect of their business, they have the characteristic that has made security problems more difficult to solve is that organizations have built large, complex computer system. In recent years, access control has been considered as an important security area. Many companies and organizations require access control mechanisms to protect important information effectively. Many researchers have developed access control models, such as discretionary access control (DAC), mandatory access control (MAC), role-based access control (RBAC), and activity-based access control (ABAC) [1-3]. The RBAC model is well known for enterprise organizations, because the main concept of RBAC is based on the enterprise environment [4-7]. RBAC has the central notion of preventing users from accessing the organization's information discretionally. Instead, access rights are associated with roles, and users are assigned to appropriate roles. RBAC is the basis of our proposed model.

---

Received October 11, 2005; revised February 20, 2006; accepted March 29, 2006.

Communicated by Jeannette Wing.

\* This research was supported by the Ministry of Information and Communication (MIC), Korea, under the Information Technology Research Center (ITRC) support program supervised by the Institute of Information Technology Assessment (IITA).

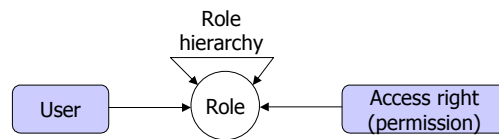


Fig. 1. Concept of RBAC model.

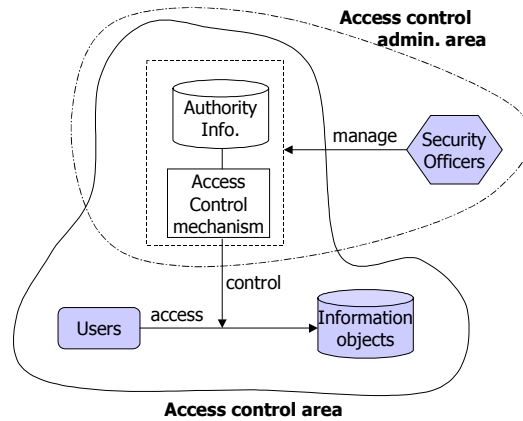


Fig. 2. Access control vs. access control administration.

**Table 1. Access control vs. security administration.**

	Access Control	Access Control Administration
Goal	control users' access	control security officers' administrative operation
Whose responsibility	local security officers	chief security officer of company
Proposed model	MAC, DAC, RBAC	ARBAC, SARBAC

Access control administration is another important issue. It includes administrative activities such as creating user accounts, managing roles, and assigning access rights to user accounts. Access control administration is a background activity, whereas access control itself is a foreground activity. Fig. 2 shows the relationship between access control and access control administration. Table 1 compares access control and access control administration.

In a large organization, access control administration is a critical issue because a single security officer cannot manage the whole access control system. Large organizations need decentralized access control administration by multiple officers. Administrative RBAC (ARBAC) and the scoped administration of RBAC (SARBAC) model have been proposed for decentralized RBAC administration [8, 9, 22]. However it has many shortcomings in practice. ARBAC uses role hierarchy as the basis of a security officer's authority range. This raises various administrative problems. We describe the shortcomings of ARBAC model in detail in section 2.

We aim to develop an effective and practical model for decentralized access control administration. Access control administration belongs to the security administration area

within the environment of role-based access control [10]. Access control administration implies: managing security officers' authority range (or administrative scope), restricting role creation, restricting role hierarchy modification, *etc.* We analyze the shortcomings of RBAC, and try to find solution for the shortcoming. As a result, we adopt different approach from ARBAC. We add a new component, the organizational structure, into the proposed model. Our proposed model's name is 'organizational structure and role-based access control (OS-RBAC)'. We will show organizational structure is a well-defined concept for user and access right management. A security officer (*SO*)'s authority range depends on the organizational unit that the *SO* belongs to. We also adopt the 'master integrity principle' (MIP), which is a basis for determining whether the *SO*'s activity is legal or illegal. The OS-RBAC model is a secure and flexible method for decentralized security administration.

The remainder of this paper is organized as follows. Section 2 presents related work. We briefly review ARBAC and related models, and describe their weaknesses. Section 3 introduces the OS-RBAC model. It contains role administration and organizational structure administration. In section 4, we compare OS-RBAC with other models, and present the advantages of the proposed model. Section 5 concludes the paper.

## 2. RELATED WORK

There is much research on role-based access control in the enterprise environment [4-7, 11-14]. As we previously mentioned, RBAC is a good access control model for a large organization. Most relational DBMS support the basic RBAC features. Although RBAC research increasingly developed, a little research has been done on RBAC administration. The most important work on RBAC administration is the ARBAC97 model. OS-RBAC has its origin in shortcomings of ARBAC. Therefore, we start by briefly describing the ARBAC97 model.

The ARBAC97 model is based on the RBAC model. Permissions are assigned to roles and users are assigned to proper roles according to their job position or role. Role hierarchy is used for efficient access rights management. Senior roles inherit junior roles' access rights in the role hierarchy. In the ARBAC97 model, administrative roles and administrative role hierarchy are added for RBAC administration. Security officers are assigned to proper administrative roles. Figs. 3 and 4 are examples of a role hierarchy and an administrative role hierarchy, respectively.

ARBAC97 has three components: URA97 for user-role assignment, PRA97 for permission-role assignment, and RRA97 for role-role assignment. URA97 uses *can-assign*( $x, y, z$ ) relation for describing security officers' assigned authority range. For example, *can-assign*(*PSO1*, *ED*, {*E1*}) means that a member of an administrative role *PSO1* can assign a user who is a current member of *ED* to be a member of the regular role *E1*. URA97 also uses *can-revoke*( $x, y$ ) for describing a revoking authority range. PRA97 has similar relations to URA97. RRA97 has more complex integrity rules for role creation/deletion and edge addition/deletion in a role hierarchy [15]. The purpose of URA97, PRA97, and RRA97 is to assign administrative authority to security officers and to prevent illegal activities by them [16]. However, ARBAC97 has many shortcomings [21]. For example, ARBAC97 allows undesirable side effect on permission-role assignment.

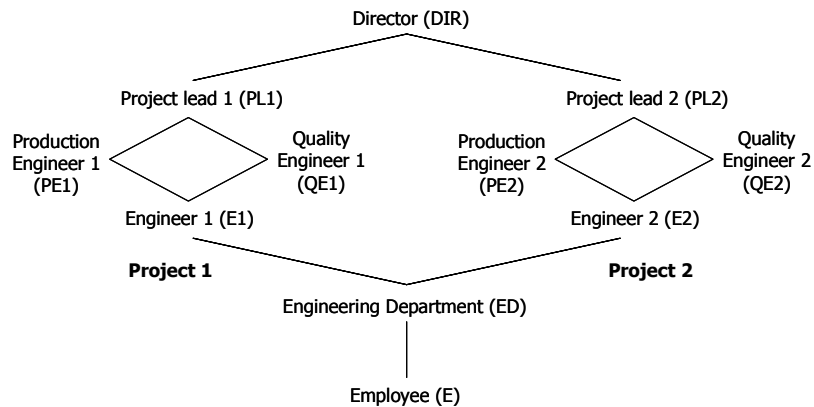


Fig. 3. An example of role hierarchy.

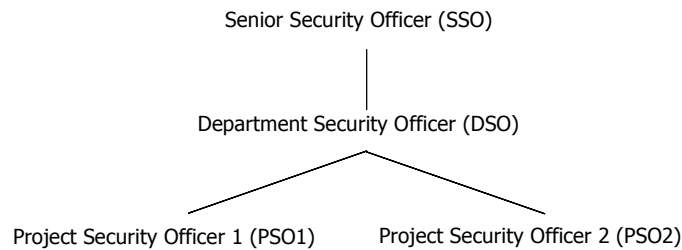


Fig. 4. An example of administrative role hierarchy.

The ARBAC02 model mitigates the shortcomings of ARBAC97 [17]. ARBAC02 moves the user/permission pool from role hierarchy to organizational structure. ARBAC02 adopts two-separated organizational structures for the user pool and the permission pool (*see note in section 4.2 for the definition of user/permission pool*). It is used into ‘prerequisite condition’s of *can-assign()* and *can-assignp()* relation. However, ARBAC02 does not solve the RRA97 problem as follows:

- unexpected side effects may take place when a security officer modifies the role hierarchy;
- ARBAC97 has complex integrity rules to prevent side effects;
- ARBAC97 allows only very simple topologies for role hierarchy;
- there is no rationale for the integrity rules;
- ARBAC97 maintains authority information data for each security officer. It is another burden for access control management.

Also, there is no comment on how to manage the administrative role hierarchy in ARBAC02.

The administrative organization based access control (AdOr-BAC) model [18] suggests using organization for access control. Or-BAC is the basis of AdOr-BAC [19, 20].

Or-BAC is a rule-based access control model. It adopts contextual rules for access control, and AdOr-BAC adds an administration function to Or-BAC. In spite of AdOr-BAC having some advantages, it does not fully use the ARBAC feature, and there is no clear principle for making contextual rules and administration functions. The contextual rules may contain defect. If the chief security officer fails to make safe rules, access control cannot be safe.

The main reason of shortcoming is that ARBAC97 defines the authority range for security officers via a role hierarchy, and security officers have the authority to modify the role hierarchy. To overcome the shortcomings of ARBAC97 and other previous work, we propose a new model named 'organizational structure and role-based access control (OS-RBAC)'.

### 3. OS-RBAC MODEL

#### 3.1 Model Summary

The OS-RBAC model is designed for decentralized security administration. It follows the basic feature of RBAC model and the administrative role/hierarchy of ARBAC97. The main idea of OS-RBAC is to add organizational structure to RBAC for security administration. An organizational structure is a hierarchy of organizational units. An organizational unit means a department such as a sales department, accounting department, or project team. Each organizational unit contains workers and authority to achieve its mission. Workers and authority are modeled as users and permissions in the OS-RBAC model. If an organization wants to apply the OS-RBAC model, it needs to follow these steps:

**Step 1:** Build an organizational structure.

**Step 2:** Allocate users/permissions to the proper organizational unit.

**Step 3:** Create administrative roles and assign security officers to them.

**Step 4:** Security officers build role or role hierarchy within their allowed authority range.

**Step 5:** Security officers assign users/permissions to roles within their authority range.

The OS-RBAC model has two submodels. Steps 1 and 2 are part of the **organizational structure administration model**, as described in section 3.2. Steps 3 to 5 are part of the **role administration model** as described in section 3.3. Fig. 5 shows the concept of the OS-RBAC model.

Main issue for decentralized security administration is how to determine each security officer's authority range. In the OS-RBAC model, membership of an organizational unit is the criterion for an authority range. If a security officer *SO1* belongs to organizational unit *OUI*, *SO1* has authority over *OUI* and its child organizational units in the organizational structure. *SO1* can handle users, permissions, and roles that belong to *OUI* and its child organizational units. Fig. 6 shows an example of authority ranges for security officers.

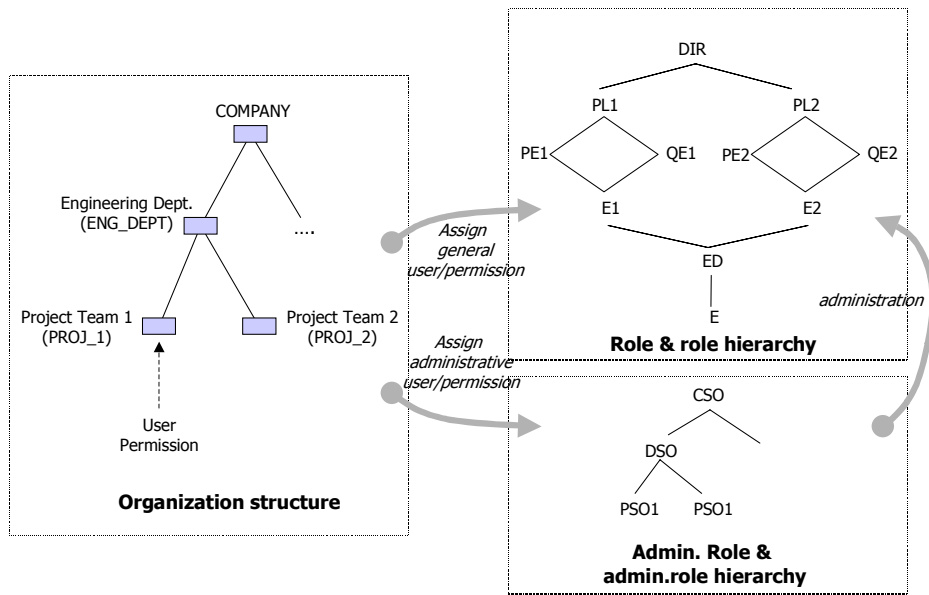


Fig. 5. Concept of OS-RBAC model.

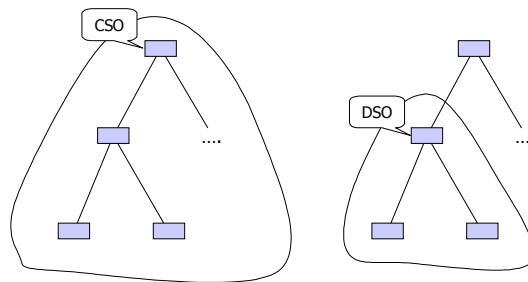


Fig. 6. An example of authority range for security officers.

Before we describe two detail models, we define common terms of the models.

**Definition 1** Basic elements of OS-RBAC.

- $OT$ : set of organizational unit  $ot$
- $U$ : set of user  $u$ 
  - $u.org\_unit$ : attribute that contains one of  $ot$
- $R$ : set of role  $r$ 
  - $r.org\_unit$ : attribute that contains one of  $ot$
  - $r.type$ : attribute that contains one of ‘G’ (general role) or ‘A’ (admin. role)
  - $r.group$ : attribute that contains one of ‘DR’ (department role) or ‘JR’ (job role)
- $P$ : set of permission  $p$ 
  - $p.org\_unit$ : attribute that contains one of  $ot$
  - $p.type$ : attribute that contains one of ‘G’ (general perm.) or ‘A’ (admin. perm.) □

The attribute *org\_unit* is used for specifying membership of user, role, and permission to an organizational unit. The attribute *group* is used for grouping roles to department role or job roles. ‘SALES\_DEPT’ and ‘ACCOUNT-DEPT’ are examples of department role, and ‘SALES\_MANAGER’ and ‘ACCOUNT\_CLERK’ are examples of job role. A department role contains common permissions for members of the department. As a result, department roles are located under job roles on a role hierarchy. In Fig. 3, *E* and *ED* may be department role, and the others may be job role.

**Definition 2** (*OS*) An organizational structure *OS* is an hierarchy of organizational unit (*ot*):  $OS \subseteq OT \times OT$ , with following properties:

- is a partially ordered tree structure
- has a maximal organizational unit named *COMPANY*
- $\forall ot \in OT, ot$  has only one direct parent on the *OS*
- $UOA \subseteq U \times OT$ : user-organization assignment between *U* and *OT*
- $POA \subseteq P \times OT$ : permission-organization assignment between *P* and *OT*

where a given user(permission) is associated with only one *OT*. □

**Definition 3** (*CSO*) Chief Security Officer *CSO* is a security officer who has whole administrative authority in the target organization. □

**Assumption 1** Before the organizational structure is generated according to the OS-RBAC rules, there exist maximal organizational unit *COMPANY* and *CSO*, and all users and permissions belong to *COMPANY*. □

### 3.2 Organizational Structure Administration Model

This model says about related to building and modifying organizational structure. If a security officer *SO1* belongs to organizational unit *OU1*, *SO1* can create a new organizational unit *OU2* under *OU1*, and can link *OU1* to *OU2* by adding an edge between them. Edge  $OU1 \rightarrow OU2$  denotes that *OU1* is a parent organizational unit of *OU2*. *SO1* also can move users/permissions from *OU1* to *OU2* (or from *OU2* to *OU1*). Finally, *SO1* can delete *OU2*. The model has administration rules for these administration activities. These rules are grouped as: UOA (user-organization assignment), POA (permission-organization assignment), and OOA (organization-organization assignment). Fig. 7 shows the groups in the organizational structure administration model.

Now we describe administration rules for the organizational structure. For simplicity we assume that *SO* is a user that has some role of type ‘A’.

#### 3.2.1 UOA (user-organization unit assignment) rules

- (1) (Deescalate a user) *SO* can deescalate a user *u* from an organizational unit *m* to an organizational unit *n* subject to:
  - $(m > n) \wedge (SO.org\_unit \geq m)$ .
 (Deescalate means that *SO* move a user *u* from a higher organizational unit *m* to a lower organizational unit *n*).

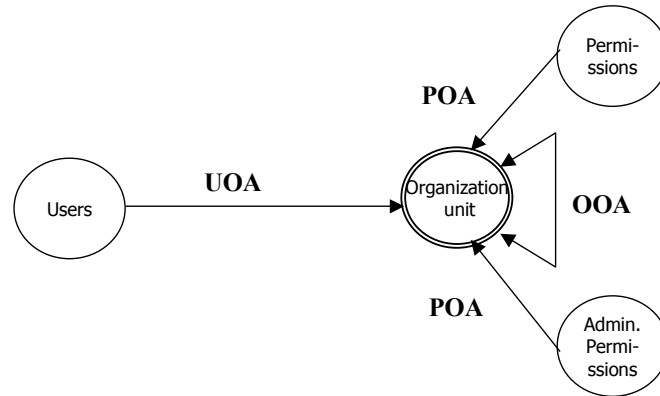


Fig. 7. Organizational structure administration model.

- (2) (Escalate a user)  $SO$  can escalate a user  $u$  from an organizational unit  $n$  to an organizational unit  $m$  subject to:  
 $-(m > n) \wedge (SO.org\_unit \geq m)$ .
- (3) (Corollary of rule (1)) If a user  $u$  is deescalated on the organizational structure, every role  $r$  that  $u$  belongs to it and  $r.org\_unit > u.org\_unit$  is revoked from  $u$ .

### 3.2.2 POA (permission-organization unit assignment) rules

- (4) (Deescalate a permission)  $SO$  can deescalate a permission  $p$  from an organizational unit  $m$  to an organizational unit  $n$  subject to:  
 $-(m > n) \wedge (SO.org\_unit \geq m)$ .
- (5) (Escalate a permission)  $SO$  can escalate a permission  $p$  from an organizational unit  $n$  to an organizational unit  $m$  subject to:  
 $-(m > n) \wedge (SO.org\_unit \geq m)$ .
- (6) (Corollary of rule (5)) If a permission  $p$  is escalated on the organizational structure,  $p$  is revoked from every role  $r$  subject to:  
 $-(p \in r) \wedge (r.group = 'JR') \wedge (r.org\_unit < p.org\_unit)$ .

### 3.2.3 OOA (organization unit-organization unit assignment) rules

- (7) (Create an organizational unit)  $SO$  can create an organizational unit with no restriction.  
 (Creation of organizational unit does not include edge insertion to any organizational unit).
- (8) (Delete an organizational unit)  $SO$  can delete an organizational unit  $m$  subject to:  
 $-(SO.org\_unit > m) \wedge (m = \phi)$ .  
 An empty ( $\phi$ ) organizational unit means that the organizational unit has no users and permissions in addition that no role's attribute of  $org\_unit$  is  $m$ . Empty unit is an un-referenced unit, and thus may be removed without affecting the model.
- (9) (Insert an edge)  $SO$  can insert an edge that makes an organizational unit  $m$  is senior to an organizational unit  $n$  subject to:  
 $-(SO.org\_unit \geq m) \wedge (n \text{ has no edge to any senior organizational unit})$ .

- (10) (Delete an edge) *SO* can delete an edge that makes an organizational unit *m* is senior to an organizational unit *n* subject to:  
 $-(SO.org\_unit \geq m) \wedge (n = \phi) \wedge (n \text{ has no junior organizational unit}).$

From the administration rules (1) to (10), we can derive the initial building process for an organizational structure. In the initial state, there is only one organizational unit *COMPANY* and one chief administrative role *CSO*. All users/permissions belong to *COMPANY*. Then *CSO* creates child organizational units under *COMPANY* and deescalates proper users/permissions to the child organizational units. *CSO* may create grandchild organizational units beneath child organizational units. If *CSO* creates administrative roles for child organizational units, they also can create grandchild organizational units and deescalate users/permissions from child organizational units to grandchild organizational units. By repeating these processes, we can build the organizational structure. It is important that we can build an organizational structure in this decentralized way.

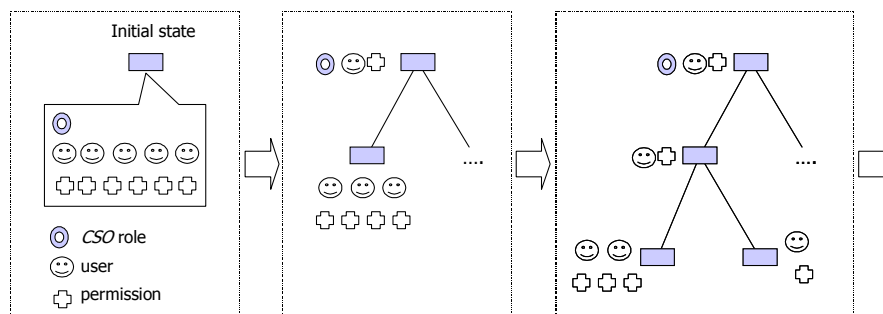


Fig. 8. The process of building an initial organizational structure.

### 3.3 Role Administration Model

The role administration model describes about administration activities related to roles for each security officer. A security officer can create/delete a role, assign users/permissions to the role, and compose part of the role hierarchy. The role administration model is composed of administration rules for the administration activities above. These rules fall into three groups, URA (user-role assignment), PRA (permission-role assignment), and RRA (role-role assignment). Fig. 9 shows the groups in the role administration model.

Now we can describe administration rules for the role administration model. For simplicity, we assume that *SO* is a security officer who belongs to administrative role *SO* ( $SO.type = 'A'$ ).

#### 3.3.1 URA (user-role assignment) rules

- (11) (Assign a user) *SO* can assign a user *u* to role *r* subject to:  
 $-(SO.org\_unit \geq u.org\_unit) \wedge (SO.org\_unit \geq r.org\_unit) \wedge (u.org\_unit \geq r.org\_unit).$

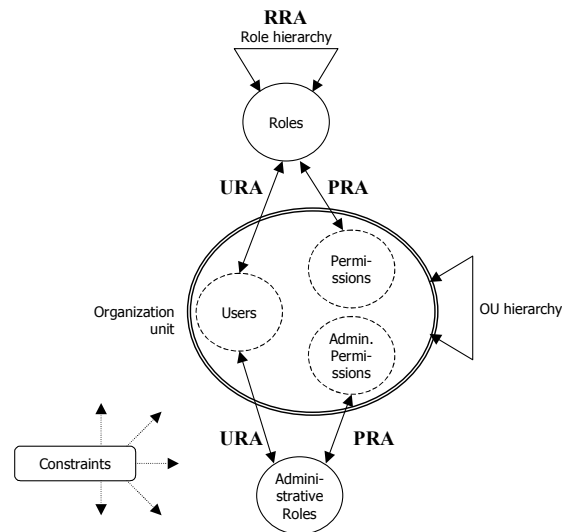


Fig. 9. Role administration model.

- (12) (Revoke a user)  $SO$  can revoke a user  $u$  from role  $r$  subject to:  
 $-(SO.org\_unit \geq u.org\_unit) \wedge (SO.org\_unit \geq r.org\_unit)$ .

### 3.3.2 PRA (permission-role assignment) rules

- (13) (Assign a permission)  $SO$  can assign a permission  $p$  to role  $r$  subject to:  
 $-(SO.org\_unit \geq p.org\_unit) \wedge (SO.org\_unit \geq r.org\_unit) \wedge (r.org\_unit \geq p.org\_unit) \wedge (r.type = p.type)$ .
- (14) (Revoke a permission)  $SO$  can revoke a permission  $p$  from role  $r$  subject to:  
 $-SO.org\_unit \geq r.org\_unit$ .

### 3.3.3 RRA (role-role assignment) rules

To define administration rules for RRA is more difficult than for the other groups. This is because composing the role hierarchy may give rise to unexpected information flow, as presented in the RRA97 model [17]. We adopt the ‘master integrity principle (MIP),’ which is presented in the previous paper [21], for preventing unexpected information flow in the role hierarchy. Here we simply describe MIP, without going into detail:

**Master Integrity Principle (MIP).** If a security officer modifies his/her authority range in the role hierarchy, it should not lead to any change of authority (access rights) of the roles that belong to any senior security officers’ authority range.

The paper [21] introduces a function *isValid()* for implementing MIP. If *isValid()* returns true, the current modification to the role hierarchy is valid. We modify this function for the OS-RBAC model. In OS-RBAC, only edge insertion/deletion leads to a

```

boolean isValid2(ar, pr, BRH, ARH) {
// ar: administrative role that makes modification
// pr: parent role in the edge insertion/deletion
// BRH: role hierarchy before a modification takes place
// ARH: role hierarchy after a modification takes place

// findNAR(ar, pr): return a role that belongs to senior administrative role of ar and is the
// nearest ancestor role of pr
// TAR(r): total access rights of role r

r = findNAR(ar, pr);
a = TAR(r) of BRH;
b = TAR(r) of ARH;
if (a ≠ b)                // if TAR(r) is changed after modification
    return false ;       // modification is invalid
else
    return true ;        // modification is valid
}

```

Fig. 10. Algorithm for implementing MIP.

change of the authority that MIP controls. The new function **isValid2**(ar, pr, BRH, ARH) is designed for when administrative role ar inserts/deletes an edge between cr and pr, where pr is a parent role of cr.

- (15) (Create a role) SO can create a role r where  $SO.type = 'A'$ . SO should specify the values of r.org\_unit, r.type, and r.group subject to:
- $SO.org\_unit \geq r.org\_unit$ .
- (16) (Delete a role) SO can delete a role r subject to:
- $(SO.org\_unit \geq r.org\_unit) \wedge (r = \phi)$ .
- (A role r is empty ( $\phi$ ) means that r has no user, permission, and edges to junior/senior role. Empty role is an un-referenced role, and thus its removal does not affect the model.)
- (17) (Add an edge) SO can add an edge that makes role x is senior to role y for four cases with conditions:
- 1)  $x.group = 'JR', y.group = 'JR'$ 
    - $(SO.org\_unit \geq x.org\_unit) \wedge (x.org\_unit \geq y.org\_unit) \wedge (\text{satisfy MIP})$ .
  - 2)  $x.group = 'JR', y.group = 'DR'$ 
    - $(SO.org\_unit \geq x.org\_unit) \wedge (\text{satisfy MIP})$ .
  - 3)  $x.group = 'DR', y.group = 'JR'$ 
    - cannot add edge
  - 4)  $x.group = 'DR', y.group = 'DR'$ 
    - $(SO.org\_unit \geq y.org\_unit) \wedge (y.org\_unit \geq x.org\_unit) \wedge (\text{satisfy MIP})$ .
- (18) (Delete an edge) SO can delete an edge that makes role x is senior to role y subject to:
- $(SO.org\_unit \geq x.org\_unit) \wedge (\text{satisfy MIP})$ .

**Note.** The role administration model does not say anything about role (de)escalation. If a security officer wants to (de)escalate, he/she may delete the role and create a new role.

**Note.** OS-RBAC model allows a junior SO to remove a user-role/permission-role/role-role association that a more senior SO has added, or to add a user-role/*etc.* that a more senior SO has removed, in cases where the user/permission/roles have org\_units below that of the junior SO. If a company wants to prevent the situation, the company may add extra attribute '*creator*' to role, URA, PRA, and RRA. If a role belongs to authority range of junior SO, the attribute '*creator*' contains name of senior SO, junior SO cannot modify the role. The principle also applies on URA, PRA, and RRA.

### 3.4 Soundness of Administration Rules

OS-RBAC comprises the organizational structure and the role administration model. The overall OS-RBAC model is shown in Fig. 11. OS-RBAC supports 18 administrative operations and each operation contains an administration rule. In this section we discuss the soundness of the administration rules. Mathematical proof of soundness is not suitable for our administrative rules. But we can trust them by following discussion:

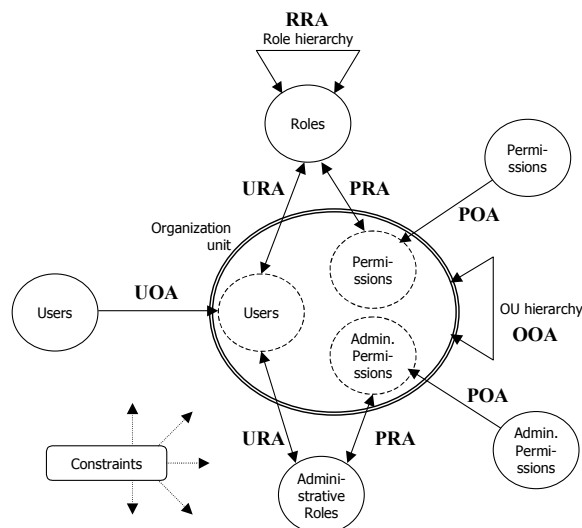


Fig. 11. Overall OS-RBAC model.

- (1) If a security administrator does security work over his/her authority range, it is false work.
- (2) A security administrator can do his/her work only by 16 administrative operations defined by OS-RBAC model. (ex. Deescalate a user, escalate a user, assign a user, *etc.*)
- (3) Each administrative operation is independently executed without interfering with the others.

- (4) Each administrative operation has one or two administration rules as a security restriction
- (5) We can specify illegal cases for each administrative operation. For example, user-role assignment operation has the three illegal cases:

- Case 1.** *SO1* tries to assign a user, who belongs to *SO1*'s authority range, to a role that does not belong to *SO1*'s authority range. (e.g.  $u1 \rightarrow r5$  in Fig. 12).
- Case 2.** *SO1* tries to assign a user, who does not belong to *SO1*'s authority range, to a role that belongs to *SO1*'s authority range. (e.g.  $u5 \rightarrow r2$  in Fig. 12).
- Case 3.** *SO1* tries to assign a user, who does not belong to *SO1*'s authority range, to a role that does not belong to *SO1*'s authority range. (e.g.  $u5 \rightarrow r5$  in Fig. 12).

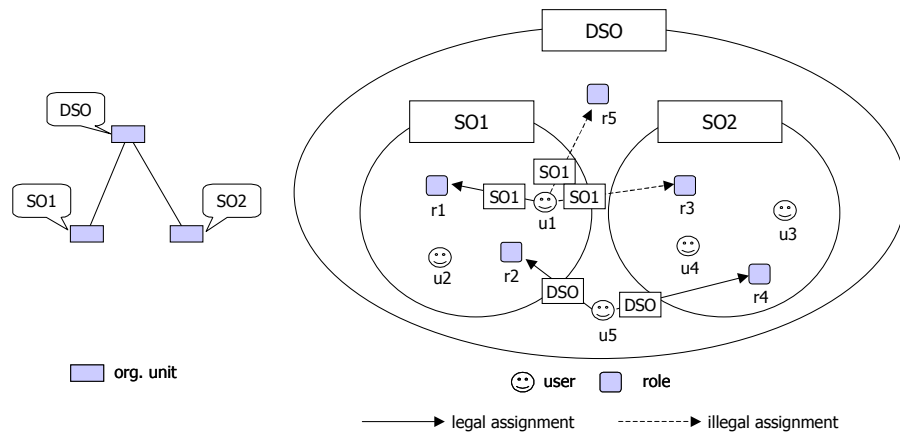


Fig. 12. An example of legal and illegal user-role assignment.

- (6) If all the illegal cases for each administrative operation conflict with related administration rule, we can accept the administration rule is sound. For example, Case 1 is prevented by the condition of administration rule (11),  $SO1.org\_unit \geq target\_role.org\_unit$ . Case 2 is prevented by the condition of the same rule,  $SO1.org\_unit \geq target\_user.org\_unit$ . Case 3 is also prevented by the condition of the same rule,  $SO1.org\_unit \geq target\_user.org\_unit \wedge SO1.org\_unit \geq target\_role.org\_unit$ . Therefore we can say that administration rule (11) is sound.

We can review the soundness of other administration rules in a similar way. For the sake of brevity, we omit a review of the soundness of the remaining administration rules.

## 4. DISCUSSION

### 4.1 Comparison with Other Models

The OS-RBAC model overcomes the shortcomings of previous models. We compare OS-RBAC with ARBAC97 and ARBAC02. Table 2 contains a summary of the

**Table 2. Comparison OS-RBAC with other models.**

	ARBAC97	ARBAC02	OS-RBAC
Base of security officer's authority range	Role hierarchy	Role hierarchy	Organization unit
User pool	Role hierarchy	Organization unit for user pool	Organization unit
Permission pool	Role hierarchy	Organization unit for permission pool	Organization unit
Checking method for illegal administrative operation (URA, PRA)	Checking allowed authority information (data)	Checking allowed authority information (data)	Checking administration rules
Checking method for illegal administrative operation (RRA)	Restrict role hierarchy structure. Checking administration rules	Restrict role hierarchy structure. Checking administration rules	Checking administration rules & MIP
Senior security officer can manage junior security officers	N/A	N/A	Yes
User, role, and permission have attributes	No	No	Yes
Maintain direct authority data for each security officer	Yes	Yes	No
Overhead of maintaining organization structure	No	Yes	Yes

Administrative role	Prerequisite condition	Role range
<i>PSO1</i>	<i>ED</i>	$[E1, PL1]$
<i>PSO2</i>	<i>ED</i>	$[E2, PL2]$
<i>DSO</i>	$ED \wedge \neg PL1$	$[PL2, PL2]$
<i>DSO</i>	$ED \wedge \neg PL2$	$[PL1, PL1]$

(a) An example of *can-assign* (user to role) information.

Administrative role	Role range
<i>PSO1</i>	$[E1, PL1]$
<i>PSO2</i>	$[E2, PL2]$
<i>DSO</i>	$[ED, DIR]$

(b) An example of *can-revoke* (user from role) information

Fig. 13. An example of administrative role's authority information.

comparison. The main difference between OS-RBAC and other models is the basis for a security officer's authority range. OS-RBAC adopts the organization unit, whereas others adopt role hierarchy. As a result, OS-RBAC is free from changes of role hierarchy. The attributes of model components are distinctive points of the proposed model. OS-RBAC uses administration rules for checking illegal administrative operations of a security officer group. Other models use authority information as shown in Fig. 13. Maintaining the

organizational structure is an overhead of OS-RBAC, but it requires little effort because the organizational structure is rarely changed.

We does not compare OS-RBAC model with rule-based model such as AdOr-BAC. Rule-based models are more expressive than RBAC model family; rule-based models intend to provide means to handle flexible and dynamic requirements. A security officer can generate necessary security rules in the rule-based context. But rule-based models have shortcomings. Firstly, the security rules generated by a security officer may have defect. It is difficult to check integrity among several security rules. Secondly, in general, rule-based security model is more difficult to implement than non-rule-based model. We think flexibility (expressiveness) of a security model is trade-off for simplicity or robustness of the model. We also think it is possible for enhancing expressiveness of OS-RBAC to adopt rule-based aspect partially, but it is out of range of the proposed model.

**Note.** The user/permission pool for an administrative role means a set of candidate users/permissions that the administrative role can choose and assign it to general roles. In the ARBAC97 model, prerequisite roles on the role hierarchy are implicit user/permission pool. In the OS-RBAC model organization units on the organization structure are explicit user/permission pool.

#### 4.2 Advantages of OS-RBAC Model

##### **OS-RBAC offers a clear criterion for role administration.**

The basis for role administration in OS-RBAC is the organizational structure. Each organizational unit has its own mission and gathers users, rights, information, and roles to achieve the mission. An organizational structure reflects the breaking down of the overall mission of an organization. A role is a group of users and access rights defined to achieve a specific mission. If a role in an organizational unit owns access rights that are not related to its mission, the access rights should be revoked. Therefore, organizational structure is a good criterion for access right management and role administration. By contrast, traditional access control models, including ARBAC97 and ARBAC02, depend on security officers' discretionary decision making for role administration.

##### **OS-RBAC does not need administrative data.**

The ARBAC97 and ARBAC02 models need administrative data, as shown in Fig. 13. Table 3 summarizes all of the required administrative data for the two models. Data-based administration has at least three weaknesses. First, there may be integrity problems in the administrative data. If a table in Table 3 contains incorrect administrative data, some security officers can perform illegal administrative operation. A more

**Table 3. Administrative data summary.**

Sub model	Tables containing required admin. data
URA	<i>can-assign, can-revoke</i>
PRA	<i>can-assignp, can-revokep</i>
RRA	<i>can-modify</i>

Administrative role	Prerequisite condition	Role range
<i>PSO1</i>	<i>ED</i>	<i>[E1, PL1]</i>
...	...	...

(a) *can-assign* data.

Administrative role	Role range
<i>PSO1</i>	<i>[E2, PL2]</i>
...	...

(b) *can-revoke* data.

Fig. 14. An example of inconsistency.

serious problem is that there is no criterion for finding incorrect administrative data; data integrity is wholly the responsibility of a senior security officer. Second, there may be inconsistencies within the administrative data. Fig. 14 shows an inconsistency between *can-assign* and *can-revoke*. Security officer *PSO1* can assign users to range  $[E1, PL1]$ , but cannot revoke users from  $[E1, PL1]$  because the *can-revoke* range is different from the *can-assign* range. Third, there may be illegal modification of administrative data.

OS-RBAC does not need such administrative data. OS-RBAC has built-in administrative rules. These rules are contained within the access control module. As a result, OS-RBAC does not have the two problems of the ARBAC models described above.

#### OS-RBAC allows flexible topologies for a role hierarchy.

Role hierarchy management is described by the RRA submodel in the ARBAC97 and ARBAC02 models. RRA defines each security officer's authority range in the role hierarchy and requires the authority range to be 'encapsulated' [8, 15]. As a result, the shape of the authority range resembles a 'net'. An encapsulated range implies that the shape of the role hierarchy cannot be a 'tree'. Fig. 15 shows possible topologies of a role hierarchy that ARBAC can support.

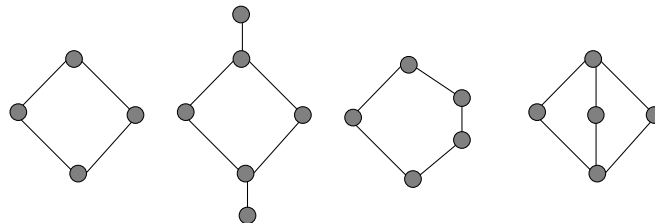


Fig. 15. Allowed topologies of role hierarchy in ARBAC model.

In the OS-RBAC model, the 'org\_unit' attribute of each role indicates which organization unit contains the role. This means that every role has its security officer. A security officer's authority range does not depend on the topology of the role hierarchy. As a result OS-RBAC supports various topologies for the role hierarchy. Fig. 16 shows topologies of a role hierarchy that OS-RBAC can support, but that ARBAC models cannot support.

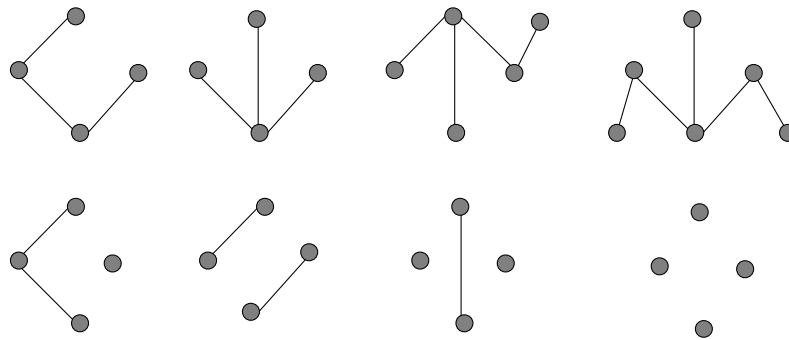


Fig. 16. Allowed topologies of role hierarchy in OS-RBAC model.

### OS-RBAC supports true decentralized access control administration.

The ARBAC model is proposed for decentralized role administration by multiple security officers. However, it seems to have centralized management of the security officer group by the chief security officer. The chief security officer creates the administrative role hierarchy, assigns users to administrative roles, and inserts administrative data into tables, as shown in Table 2. In the OS-RBAC model, each security officer has authority to create a junior security officer's role. He/she also has authority to create junior organization units under his/her own organization unit. This means that OS-RBAC can support true decentralized role administration.

## 5. CONCLUSION

In this paper, we present a new role administration model using organizational structure. Large organizations require decentralized role administration. A core task for decentralized role administration is to define each security officer's authority range. The ARBAC97 and ARBAC02 models define authority range via the role hierarchy. This leads many shortcomings. Our proposed model, OS-RBAC, defines authority range via the organizational structure, which is separated from the role hierarchy. OS-RBAC adopts internal administrative rules instead of administrative data. OS-RBAC supports flexible topologies for a role hierarchy, and true decentralized role administration.

Organizational structure is a good basis for authority management. Our organizational structure can combine with other access control models such as mandatory access control (MAC), discretionary access control (DAC), and purposed-based access control. Delegation in OS-RBAC is another research topic, and we will consider the modification of OS-RBAC for various organizations such as banks, governments, universities, *etc.*

## REFERENCES

1. E. G. Amoroso, *Fundamental of Computer Security Technology*, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1994.
2. C. P. Pfleger, *Security in Computing*, Prentice-Hall International Inc., 2nd ed., New Jersey, 1997.

3. D. Russel and G. T. Gangemi Sr., *Computer Security Basics*, O'Reilly and Associates, Inc., 1991.
4. E. Barka and R. Sandhu, "Framework for role-based delegation models," in *Proceedings of the 14th Annual Computer Security Applications Conference*, 2000, pp. 168-176.
5. D. Ferraiolo, J. Cugini, and R. Kuhn, "Role-based access control (RBAC): features and motivations," in *Proceedings of the 11th Annual Computer Security Application Conference*, 1995, pp. 241-248.
6. R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control method," *IEEE Computer*, Vol. 29, 1996, pp. 38-47.
7. R. Sandhu, D. Ferraiolo, and D. Kuhn, "The NIST model for role-based access control: towards a unified standard," in *Proceedings of the 5th ACM Workshop on Role-Based Access Control*, 2000, pp. 47-63.
8. R. Sandhu, V. Bhamidipati, and Q. Munawar, "The ARBAC97 model for role-based administration of roles," *ACM Transactions on Information and System Security*, Vol. 2, 1999, pp. 105-135.
9. R. Sandu and Q. Munawar, "The ARBAC99 model for administrative roles," in *Proceedings of the 15th Annual Computer Security Applications Conference*, 1999, pp. 229-240.
10. E. B. Fernandez and J. C. Hawkins, "Determining role rights from use cases," in *Proceedings of the 2nd ACM Workshop on Role-Based Access Control*, 1997, pp. 121-125.
11. S. Oh and S. Park, "Task-role-based access control model," *Information Systems*, Vol. 28, 2003, pp. 533-562.
12. S. Oh and S. Park, "Enterprise model as a basis of administration on role-based access control," in *Proceedings of the 3rd International Symposium on Cooperative Database Systems for Advanced Applications*, 2001, pp. 165-174.
13. R. K. Thomas, "Team-based access control (TMAC): a primitive for applying role-based access controls in collaborative environments," in *Proceedings of the 2nd ACM Workshop on Role-Based Access Control*, 1997, pp. 13-19.
14. J. D. Moffett and E. C. Lupu, "The use of role hierarchies in access control," in *Proceedings of the 4th ACM Workshop on Role-Based Access Control*, 1999, pp. 153-160.
15. R. Sandhu and Q. Munawar, "The RRA97 model for role-based administration of role hierarchies," in *Proceedings of the 13th Annual Computer Security Application Conference*, 1998, pp. 39-49.
16. S. Oh and R. Sandhu, "A model for role administration using organization structure," in *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies*, 2002, pp. 155-162.
17. F. Cuppens, P. Balbiani, S. Benferhat, Y. Deswarte, A. Abou El Kalam, R. Elbaida, A. Mige, C. Saurel, and G. Trouessin, "Organization based access control," in *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, 2003, pp. 120-130.
18. F. Cuppens and A. Mige, "Modeling contexts in the or-BAC model," in *Proceedings of the 19th Applied Computer Security Associates Conference*, 2003, pp. 416-427.
19. F. Cuppens and A. Mige, "Administration model for or-BAC," in *Workshop on Metadata for Security, International Federated Conference*, 2003, pp. 754-768.

20. S. Oh, "Master integrity principle for effective management of role hierarchy," *Journal of Korea Information Processing Society*, Vol. 12-C, 2005, pp. 981-988.
21. J. Crampton and G. Loizou, "Administrative scope: a foundation for role-based administrative models," *ACM Transactions on Information and System Security (TISSEC)*, Vol. 6, 2003, pp. 201-231.



**Sejong Oh (吳世宗)** earned his Ph.D. in the Department of Computer Science from Sogang University, Seoul, Korea, in 2001. He worked as a Post Doctor researcher of the School of Information Technology and Engineering, George Mason University, U.S.A. He has been a professor of Computer Science Department, Dankook University, Cheonan, Korea, since 2003. His main research interests include access control for enterprise and distributed systems, ERP, secure DBMS, and Ubiquitous security. Dr. Oh is a member of Korea Information Processing Society and Korea Institute of Information Security and Cryptology.



**Changwoo Byun (邊昶又)** received the B.S. and M.S. degrees in the Department of Computer Science from Sogang University, Seoul, Korea, in 1999 and 2001, respectively. Since 2001, he has been studying for a Ph.D. at the Department of Computer Science of Sogang University. His areas of research include role-based access control model, access control for distributed systems, access control for XML data, Dynamic access control, and Ubiquitous security.



**Seog Park (朴錫)** is a Professor of Computer Science at Sogang University. He received the B.S. degree in Computer Science from Seoul National University in 1978, the M.S. and the Ph.D. degrees in Computer Science from Korea Advanced Institute of Science and Technology (KAIST) in 1980 and 1983, respectively. Since 1983, he has been working in the Department of Computer Science of the College of Engineering, Sogang University. His major research areas are database security, real-time systems, data warehouse, digital library, multimedia database systems, role-based access control and Web database. Dr. Park is a member of the IEEE Computer Society, ACM and the Korea Information Science Society. Also, he has been a member of Database Systems for Advanced Application (DASFAA) steering committee since 1999.