Authorization Algorithms for the Mobility of User-Role Relationship

Hua Wang [1] **Lili Sun** [1] **Yanchun Zhang** [2] **Jinli Cao** [3]

[1] Department of Maths & Computing, University of Southern Queensland
Toowoomba QLD 4350 Australia
Email: (wang, sun)@usq.edu.au
[2] School of Computer Science and Mathematics
Victoria University of Technology, Melbourne City, MC8001, Australia.
Email: yzhang@csm.vu.edu.au
[3] Department of Computer Science & Computer Engineering
La Trobe University, Melbourne, VIC 3086, Australia
Email: jinli@cs.latrobe.edu.au

Abstract

The mobility of user-role relationship is a new feature relative to their counterparts in user-role assignments. When an administrative role assigns a role to a user with a mobile membership, this allows the user to use the permissions of the role and to be further added other roles by administrators. Immobile membership grants the user the authority to use the permissions, but does not make the user eligible for further role assignment. Two types of problems may arise in user-role assignment with the mobility of user-role relationship. One is related to authorization granting process. When a role is granted to a user, this role may be conflict with other roles of the user or together with this role; the user may have or derive a high level of authority. Another is related to authorization revocation. When a role is revoked from a user, the user may still have the role from other roles.

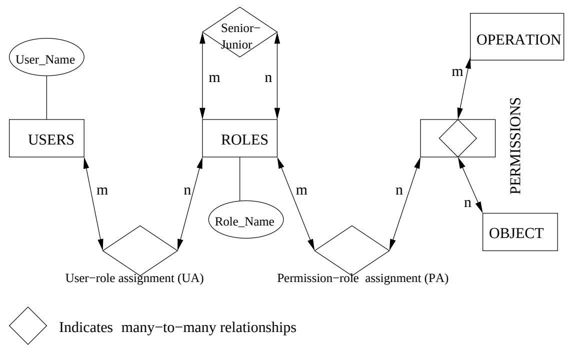
In this paper, we discuss granting and revocation models related to mobile and immobile memberships between users and roles, then provide proposed authorization granting, weak revocation and strong revocation algorithms that are based on relational algebra and operations. We also describe how to use the new algorithms with an anonymity scalable payment scheme. Finally, comparisons with other related work are made.

Keywords: Authorization, mobility, RBAC, User-Role Relationship.

1 Introduction

The National Institute of Standards and Technology (NIST) developed role-based access control (RBAC) prototype (Feinstein H. L. 1995) and published a formal model (Ferraiolo D. F. and Kuhn D. R. 1992). Many organizations prefer to centrally control and maintain access rights, not so much at the system administrator's personal discretion but more in accordance with the organization's protection guidelines (David F. F., Dennis M. G. and Nickilyn L. 1993). RBAC is being considered as part of the emerging SQL3 standard for database management systems, based on their implementation in Oracle 7 (Sandhu R. 1998*b*). Many RBAC practical applications have been implemented (Barkley J. F., Beznosov K. and Uppal J. 1999, Ferraiolo D. F., Barkley J. F. and Kuhn D. R. 1999, Sandhu R. 1998*a*).

RBAC involves individual users being associated with roles as well as roles being associated with permissions (Each permission is a pair of objects and operations). As such, a role is used to associate users and permissions. A user in this model is a human being. A role is a job function or job title within the organization associated with authority and responsibility. As shown in Figure 1, the relationships between users and roles, and between roles and permissions are many-to-many. Roles are in two categories within RBAC, one is administrative roles (admin.role), the other is regular roles (role) that need to be assigned to or revoked from users by administrative roles.



Senior−Junior
User_Name
m
n
OPERATION
m
USERS
ROLES
PERMISSIONS
m
n
m
n
n
Role_Name
OBJECT
User−role assignment (UA)
Permission−role assignment (PA)
Indicates many−to−many relationships

Figure 1: RBAC relationship

There may be hierarchies within roles. Senior roles are shown at the top of the hierarchies. Senior roles inherit permissions from junior roles. Let $x > y$ denote x is senior to y with obvious extension to $x \geq y$. We use administrative roles in Figure 2 to explain the mobility of user-role membership. The Figure shows hierarchies of administrative roles. The senior administrative role (SADrole) inherits administrative role (ADrole) with all permissions of ADrole. Similarly, ADrole inherits junior administrative role (JADrole). When a membership between user and role is mobile (immobile), we say the user is a mobile (immobile) member of the role or the role is a mobile (immobile) member of the user. Supposing ADrole assigns a role to a user as a mobile member, it means the user can use permissions of the role and other administrative roles (SADrole, JADrole) may further assign other roles to the user. If the user is assigned a role as an immobile member by ADrole, it grants the user the authority to use permissions of that role and cannot be assigned other roles to the user except the immobile relationship is removed.

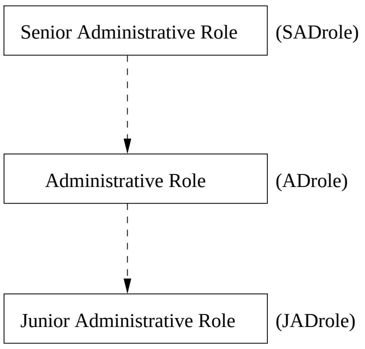


Figure 2: Hierarchies of administrative roles

However, there is a consistency problem when using RBAC management (Sandhu R. and Park J. S. 1998). For instance, if there are hundreds of roles and thousands of users in a system, it is very difficult to maintain consistency because it may change user's authorization level, or imply high-level confidential information when more than one role is requested and granted.

In this paper, we analyze authorization granting and revocation models with the mobility of user-role assignment. Then we develop authorization algorithms to check conflicts and therefore help allocate the roles without compromising the security. The algorithms are based on relational structure, relational algebra, which has a set of complete and sound axioms. As far as we know, this is the first kind of work in this area to provide authorization algorithms for role allocation and conflict detection within mobility of memberships.

The paper is organized as follows. In the next section, we consider the mobility of user-role relationship and the problems related to role assignment and revocation. An authorization granting algorithm and revocation algorithms with relational algebra are developed in section 3. In section 4, we review an anonymity scalable electronic commerce payment scheme and then apply the formal authorization approaches to this scheme. Comparisons with related work are discussed in section 5 and the conclusions are in section 6.

2 Motivation and problem definitions

Two issues are needed to address in user-role assignment. The first is to control the roles that an administrative role has authority over. Figure 3, for example, shows the administrative role (ShopSO) which co-exists with roles SHOP, SELLER etc. ShopSO controls roles say SHOP, SELLER, and AUDITOR in a shop. The second is to control which users are eligible with membership of these roles.

There are two kinds of membership between users and roles, namely mobile and immobile. When a user is assigned a role with mobile membership by an administrative role, the user is authorized to use the permissions of that roles and he/she is eligible to further accept other roles. A user with an immobile membership of a role only gets the permissions of that role and cannot be assigned other roles.

This distinction between mobile and immobile memberships can be very important in practice (Oh S. and Sandhu R. 2002, Sandhu R. and Munawer Q. 1999). For example, in a University, a guest can be granted immobile membership in a role as an observer by an administrative role but cannot be granted a staff role by administrative roles. Therefore, the guest

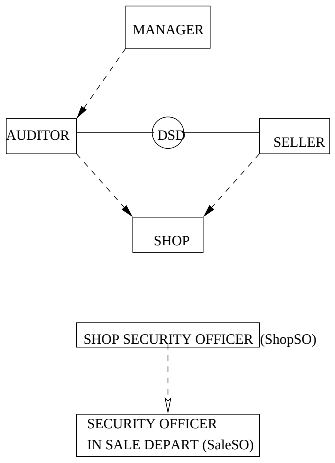


Figure 3: Administrative role and role relationships in a shop

is able to visit the University but cannot access students' data as a staff member. This can improve security of the system in the University. Another example is in a shop model mentioned above, a person under training can be assigned role SHOP as an immobile member by ShopSO and thus participate in the shop while preventing from assigning other roles (e.g. SELLER and AUDITOR) to the person. After completion of training the membership of the person can be upgraded to be mobile by ShopSO. Another example using the shop is a consultant who may grant the SHOP role as an immobile member by ShopSO. The consultant can participate in the shop and use the general resources of the shop. At the same time the immobility prevents from assigning SELLER or AUDITOR to the consultant.

These examples demonstrate that there are a lot of situations with mobile and immobile relationships between users and roles in practice. However, there are two problems that may arise in user-role assignment. They are:

Authorization granting problem – How to check whether a role is in conflict with the roles of a user when granting the role to the user as mobile or immobile member?

Authorization revocation problem – How to find whether or not a role with mobile or immobile membership of a user has been revoked from the user?

For example, Figure 3 shows the management of roles SHOP, SELLER, AUDITOR and MANAGER by ShopSO and its junior role SaleSO. Role MANAGER inherits SELLER with its permissions. SELLER has DSD relationship with AUDITOR where DSD stands by dynamic separated duty that is a constraint specifying that a user can be authorized for two different roles but cannot act simultaneously in both (Wang H., Cao J. and Zhang Y. 2002). ShopSO can assign a user to be AUDITOR or SELLER but not both simultaneously, otherwise it compromises the security of the shop system. It is easy to find conflicts between roles when assigning roles to a user in a small database system but it is hard to find them when there are thousands of roles in a system. Our aim is to provide relational algebra algorithms to solve such problems and then automatically check conflicts when assigning and revoking. For convenience, we recall some basic definitions from paper (Wang H., Cao J. and Zhang Y. 2002) with no further explanation.

Let D be a database with a set of relations REL, and a set of attributes A. REL includes ROLES, USERS, Can-assign, Can-revoke, SEN-JUN, and Role-User etc. A includes attributes such as RoleName, UserID, UserName etc from the relations. R_1 is the set of roles $R_1 = \{r_1, r_2, ..., r_n\}$; U is the set of users $U = \{u_1, u_2, ..., u_l\}$.

Some relations in set REL are detailed below.

ROLES - This relation has $(n+1)$ attributes when there are n roles.

The first attribute, RoleName is the primary key for the relation, and represents the name of a role. From the second attribute to $(n+1)th$ attribute, it describes the state of conflicts with the RoleName in the relation and its domain is {-1, 0}, where '-1' means conflicting with the RoleName and '0' means not.

The ROLES relation in Figure 3 is in Table 1. The attribute SELLERC shows whether the role SELLER is conflicting with the RoleName in the relation or not. For instance, in the third tuple, a user with role SELLER has conflicts with the role AUDITOR.

RoleName	MANAC	AUDC	SELLERC	SHOPC
MANAGER	0	0	0	0
AUDITOR	0	0	-1	0
SELLER	0	-1	0	0
SHOP	0	0	0	0

Table 1: The relation ROLES in Figure 3

Note that it is not necessary to distinguish mobile and immobile roles in the ROLES relation.

USERS - This relation has two attributes {UserID, UserName }, UserID is the identity of a user and UserName, which domain is the set of a list of users in the system. UserID and UserName are recorded for each user. UserName is the primary key for the table. For example, there are two users David and Tony in the system of Figure 3. The USERS relation is in Table 2.

UserID	UserName
0001	David
0002	Tony

Table 2: The relation USERS in Figure 3

SEN-JUN - This is a relation of roles in a system. Senior is the senior of the two roles. Table 3 expresses the SEN-JUN relationship in Figure 3.

Senior	Junior
MANAGER	AUDITOR
MANAGER	SELLER
MANAGER	SHOP
SELLER	SHOP
AUDITOR	SHOP

Table 3: SEN-JUN table in Figure 3

ROLE-USER - Defines a relationship between USERS and ROLES {RoleName, UserName}.

RoleName is a foreign key RoleName in the ROLES. It explains a role is assigned to a user.

UserName is a foreign key UserName from the USERS.

Suppose MANAGER is assigned to user Tony and SELLER to user David, Table 4 expresses the ROLE-USER relationship.

It is easy to know a role set associated with a user from ROLE-USER. Therefore, the authorization granting problem can be changed to whether a role is conflicting with a set of roles R or not.

RoleName	UserName
MANAGER	Tony
SELLER	David

Table 4: ROLE-USER table

3 Authorization granting and revocation algorithms based on relational algebra

We now propose granting and revocation algorithms based on relational algebra. A user's membership in a role can be mobile or immobile. Mobile membership of user u in role x means that u can use permissions of the role x and u can also be put into other roles. Immobile membership of user u in role x means that u can use permissions of role x but cannot be put into other roles. Each role x is separated into two subroles Mx and IMx. Membership in Mx is mobile while membership in IMx is immobile. Assignment of Mx to a user specifies that the user has a mobile membership of x. Similarly, assignment of IMx to a user specifies that the role x is an immobile member of u.

The explicit members of a role x is the set of users $\{U|(U,x) \in UA\}$ and the implicit members of role x is the set of users $\{U|\exists x' > x, (U,x') \in UA\}$ where UA is user-role assignment. Based on the mobile and immobile membership with the notion of explicit and implicit membership, there are four kinds of user-role membership for a given role x (Sandhu R. and Munawer Q. 1999).

1. *Explicit Mobile Member EMx*
   $EMx = \{u, |(u, Mx) \in UA\}$
2. *Explicit Immobile Member EIMx*
   $EIMx = \{u, |(u, IMx) \in UA\}$
3. *Implicit Mobile Member ImMx*
   $ImMx = \{u, |\exists x' > x, (u, Mx') \in UA\}$
4. *Implicit Immobile Member ImIMx*
   $ImIMx = \{u, |\exists x' > x, (u, IMx') \in UA\}$

A user may have all four kinds of membership in a role at the same time. However, we limit strict precedence amongst these four kinds of membership as follows:

$$EMx > EIMx > ImMx > ImIMx$$

Therefore only one of the membership is actually in effect at any time even though a user has multiple kinds of membership in a role. Before introducing use-role assignment with mobile and immobile relationships we interpret the concept of prerequisite conditions.

A **prerequisite condition** is evaluated for a user u by interpreting role x to be true if

$$u \in EMx \vee (u \in ImMx \wedge u \notin EIMx)$$

and $\bar{x}$ to be true if

$$u \notin EMx \wedge u \notin EIMx \wedge u \notin ImMx \wedge u \notin ImIMx$$

In other words x denotes mobile membership (explicit or implicit) and $\bar{x}$ denotes absence of any kind of membership.

For a given set of roles R let CR denote all possible prerequisite conditions that can be formed using the roles in R. Not every administrator can assign a role to a user. The following relations provide what roles an administrator can assign mobile members or immobile members with prerequisite conditions.

Can-assign-M is a relation of $\subseteq AR \times CR \times 2^R$, which is used for user-role assignments as mobile members, where AR is a set of administrative roles. On the other hand, user-role assignments as immobile members are authorized by the relation **Can-assign-IM** $\subseteq AR \times CR \times 2^R$ ⋄

User-role assignment (UA) is authorized by *Can-assign-M* and *Can-assign-IM* relations. Table 5 and Table 6 shows the *Can-assign-M* and *Can-assign-IM* relations with the prerequisite conditions in the example.

Admin.role	Prereq.Condition	Role Range
ShopSO	$\emptyset$	{SHOP}
ShopSO	SHOP $\wedge\overline{SELLER}$	{AUDITOR}
ShopSO	SHOP $\wedge\overline{AUDITOR}$	{SELLER}

Table 5: Can-assign-M in the example

Admin.role	Prereq.Condition	Role Range
ShopSO	$\emptyset$	{SHOP}

Table 6: Can-assign-IM in the example

The meaning of *Can-assign-M* $(ShopSO, SHOP, \{SELLER, AUDITOR\})$ is that ShopSO can assign a user whose current membership satisfies the prerequisite condition SHOP to be a mobile member of roles SELLER and AUDITOR. Whereas the meaning of *Can-assign-IM* $(ShopSO, SHOP, \{SELLER, AUDITOR\})$ is that ShopSO can assign a user whose current membership satisfies the prerequisite condition SHOP to be an immobile member of roles SELLER and AUDITOR. To identify a role range within the role hierarchy, the following closed and open interval notation is used.

$$[x, y] = \{r \in R | x \geq r \wedge r \geq y\}$$
$$(x, y] = \{r \in R | x > r \wedge r \geq y\}$$
$$[x, y) = \{r \in R | x \geq r \wedge r > y\}$$
$$(x, y) = \{r \in R | x > r \wedge r > y\}$$

Supposing an administrator role (ADrole) wants to assign a role r_j to user u with a set of roles R which may include mobile and immobile members. The roles r_j cannot assign to u if there is an immobile role in R. It is easy to check whether an immobile role exists in R or not. Therefore we only discuss R without any immobile role. The role r_j may be assigned as a mobile or immobile member if there is no conflicts between r_j and the roles in R. We discuss both mobile and immobile members in the following algorithm. The structure of the algorithm is shown in Figure 4. R^* is an extension of R, $R^* = \{x | x \in R\} \cup \{x | \exists x' \in R, x' > x\}$.

We use algebra operations *selection* σ and *projection* π in the algorithm.

Authorization granting algorithm

```
Grant(ADrole, $R, r_j$)
Input: ADrole, a set of roles $R$ and a role $r_j$.
Output: true iff ADrole can assign role $r_j$ to $R$ with no conflicts; false otherwise.
Begin:
Step 1. /* Whether the ADrole can assign the role $r_j$ to $R$ or not */
```

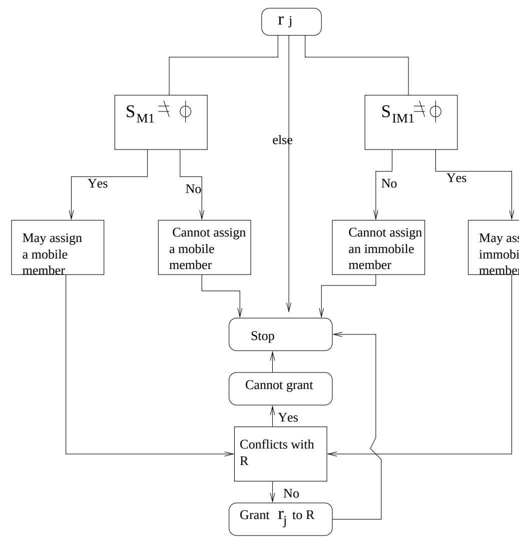

r j
$S_{M1} \neq \phi$
$S_{IM1} \neq \phi$
else
Yes
No
No
Yes
May assign a mobile member
Cannot assign a mobile member
Cannot assign an immobile member
May assign an immobile member
Stop
Cannot grant
Yes
Conflicts with R
No
Grant r_j to R

Figure 4: Grant algorithm structure

Suppose $S_{M1} = R^* \cap S_{M2}$ and $S_{IM1} = R^* \cap S_{IM2}$ where

```
$S_{M2} = \pi_{Prereq.Condition}(\sigma_{admin.role=ADrole}(Can{-}assign{-}M))$
/*$S_{M2}$ is the set of prerequisite conditions with ADrole in Can-assign-M */
$S_{IM2} = \pi_{Prereq.Condition}(\sigma_{admin.role=ADrole}(Can{-}assign{-}IM))$
/*$S_{IM2}$ is the set of prerequisite conditions with ADrole in Can-assign-IM */
Suppose $r_j$ will be a mobile member of the user,
if $S_{M1} \neq \phi$, and there exists role $r \in S_{M1}$, such that
    $r_j \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can - assign - M))$
    go to step 2
    /* $r_j$ is in the range to be assigned as a mobile
    member by ADrole in Can-assign-M, where
    $\sigma_{\{ADrole,r\}}$ is an abbreviation for
    admin.role = ADrole $\wedge$ prereq.condition = r */
    Suppose $r_j$ will be an immobile member of $u$,
if $S_{IM1} \neq \phi$ and there exists role $r \in S_{IM1}$, such that
    $r_j \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can - assign - IM))$
    go to step 2
    /* $r_j$ is in the range to be assigned as an immobile
    member by ADrole in Can-assign-IM */
    else
    return false and stop.
    /*the admini.role has no right to assign the role
    $r_j$ as a mobile or immobile member to $R$*/

Step 2. /*whether the role $r_j$ is conflicting with roles
in R or not*/

    for each role $r_i$ in $R$, do
    $Num_i = \pi_{r_jC}(\sigma_{RoleName=r_i}(ROLES))$

    /* $r_jC$ is an attribute that describes conflicting
    states of $r_j$ with other roles */
    if $Num_i = -1$
    $r_j$ is a conflicting role with $R$, return false;
    if for all $r_i \in R$, $Num_i = 0$
    return true.
    /* $r_j$ is not a conflicting role with R */          ⋄
```

This algorithm provides a way for whether a role can be assigned as mobile or immobile member to a user. For mobile membership, S_{M1} cannot be empty while for immobile membership, S_{IM1} cannot be empty.

Admin.role	Prereq.ConditionR	Role Range
ShopSO	SHOP	[SHOP, MANAGER)

Table 7: Can-revoke-M

Admin.role	Prereq.ConditionR	Role Range
ShopSO	SHOP	{SHOP}

Table 8: Can-revoke-IM

Theorem 1 The authorization granting algorithm provides a solution for the authorization granting problem. It can prevent conflicts when assigning a role to a user with mobile and immobile memberships.

Proof Assuming ADrole wants to assign a role r_j as a mobile member to a user which associates with a role set R. While step 1 in the algorithm has checked whether the ADrole can assign the role r_j as a mobile member to the user or not, the second step has decided whether the role r_j is conflicting with roles in R or not. Indeed, r_j can be assigned to the user if for all $r_i \in R$, $Num_i = 0$. Otherwise r_j is a conflicting role with R. $\diamond$

The authorization granting problem is solved by the authorization granting algorithm. Now we consider revocation of user-role membership. Similar to the *Can-assign-M* and *Can-assign-IM* relations, there are *Can-revoke-M* and *Can-revoke-IM* relations between administrative roles and regular roles.

The evaluation of a prerequisite condition for the revoke model is different from the grant model. In the revoke model a **revoke prerequisite condition** is evaluated for a user u by interpreting role x to be true if

$$u \in EMx \vee u \in EIMx \vee u \in ImMx \vee u \in ImIMx$$

and $\bar{x}$ to be true if

$$u \notin EMx \wedge u \notin EIMx \wedge u \notin ImMx \wedge u \notin ImIMx$$

In our previous paper (Wang H., Cao J. and Zhang Y. 2002) the relation $Can-assign$ involves the prerequisite conditions but $Can-revoke$ does not. The prerequisite conditions in revocation is also important. For example, if a user Bob is a member of SHOP then ShopSO may assign Bob to any role (mobile or immobile) of the shop, namely SHOP, SELLER and AUDITOR. These assignments are governed by the relations *Can-assign-M* and *Can-assign-IM*. Suppose ShopSO does not like Bob to be a member of any role outside the shop. If Bob is assigned a role that falls outside the shop then ShopSO needs revoke him from that role. Prerequisite conditions in can-revoke are one means to provide this function.

Relations **Can-revoke-M** $\subseteq AR \times PCR \times 2^R$ and **Can-revoke-IM** $\subseteq AR \times PCR \times 2^R$ show which role range of mobile membership and immobile membership administrative roles can revoke respectively, where AR is the set of administrative roles and PCR is the set of revoke prerequisite conditions. $\diamond$

The meaning of *Can-revoke-M* $(ShopSO, SHOP, [SHOP, MANAGER))$ in Table 7 is that ShopSO can revoke mobile membership of a user from any role in [SHOP, MANAGER) who satisfies the revoke prerequisite condition SHOP. Similarly, the *Can-revoke-IM* in Table 8 refers to immobile membership.

Due to role hierarchy, a role x' has all permissions of role x when x' inherits x. A user with two roles $\{x', x\}$ still has the permissions of x if only x is revoked from the user. To solve the authorization revocation problem, we need to revoke the explicit member of a role first if a user is an explicit member, then revoke the implicit member.

Following are two algorithms for revocation of role r_j from a user u by ADrole, where R is the set of roles which have been assigned to the user u. The first one is a weak revocation algorithm and another one is a strong one. The weak revocation only revokes explicit mobile or immobile memberships from u and does not revoke implicit mobile and immobile memberships but the strong algorithm revokes both explicit and implicit mobile and immobile members. The structure of weak revocation algorithm is shown in Figure 5.

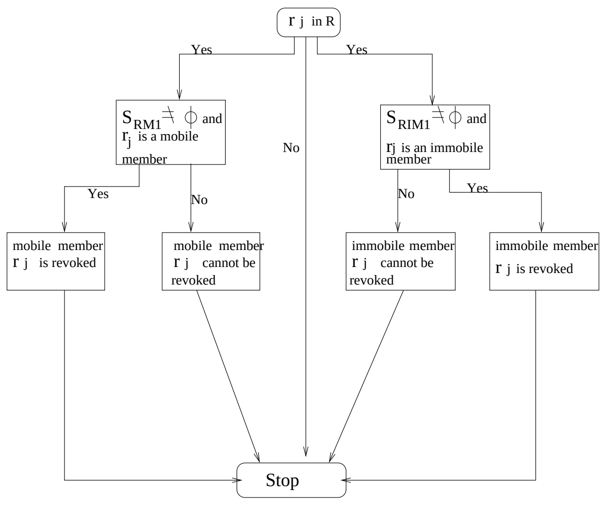

r j in R
Yes
Yes
S_RM1 ≠ φ and r j is a mobile member
No
S_RIM1 ≠ φ and rj is an immobile member
Yes
No
No
Yes
mobile member r j is revoked
mobile member r j cannot be revoked
immobile member r j cannot be revoked
immobile member r j is revoked
Stop

Figure 5: Structure of weak revocation algorithm

Weak revocation Algorithm

```
Weak_revoke(ADrole, $R$, $r_j$)
Input: ADrole, a set of roles $R$ and a role $r_j$.
Output: true if ADrole can weakly revoke role $r_j$ from $R$; false otherwise.
Begin:
Step 1:
    if $r_j \notin R$,
    return false and stop;
    /* there is no effect with the operation of the weak
    revocation since the user is not an explicit
    member of $Mr_j$*/

Step 2. /* Whether the ADrole can revoke the
    role $r_j$ from R or not */

Suppose
$S_{RM1} = R^* \cap S_{RM2}$ and $S_{RIM1} = R^* \cap S_{RIM2}$,
where

$S_{RM2} = \pi_{Prereq.ConditionR}(\sigma_{admin.role=ADrole}(Can-revoke-M))$

/*Prerequisite conditions associated ADrole in Can-revoke-M*/

$S_{RIM2} = \pi_{Prereq.ConditionR}(\sigma_{admin.role=ADrole}(Can-revoke-IM))$

/*Prerequisite conditions associated ADrole in Can-revoke-IM*/

Suppose $u \in EMr_j$ ($u$ is an explicit mobile member of $r_j$)
if $S_{RM1} \neq \phi$ and there exists role $r \in S_{RM1}$, such that
    $r_j \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can-revoke-M))$
    Return True
    /* $r_j$ is in the range to be revoked by ADrole in
    Can-revoke-M */
Suppose $u \in EIMr_j$ ($u$ is an explicit immobile member of $r_j$)
if $S_{RIM1} \neq \phi$ and there exists role $r \in S_{RIM1}$, such that
    $r_j \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can-revoke-IM))$
    Return True
    /* $r_j$ is in the range to be revoked by ADrole in
    Can-revoke-IM and the immobile membership
    is revoked */
    else
    return false and stop.
    /*the admini.role has no right to revoke the role
    $r_j$ from u */                                   ◇
```

The weak revocation algorithm can be used to check whether an administrator can weakly revoke mobile and immobile memberships from users or not.

Theorem 2 Roles r_j as mobile or immobile member is revoked by the weak revocation algorithm $Weak\_revoke(ADrole, R, r_j)$ if the user is an explicit member of role r_j and the ADrole has the right to revoke r_j from the *Can-revoke-M* and *Can-revoke-IM* relations. ◇

A user still has permissions of a role which has been weakly revoked if a role associated with the user is senior to the role revoked. To solve the authorization revocation problem, we need strong revocation which requires revocation of both explicit mobile and immobile memberships and implicit mobile and immobile memberships. Strong revocation of a user's mobile and immobile memberships in x requires the removal of not only from explicit mobile and immobile memberships in x, but also from explicit and implicit mobile and immobile memberships in all roles senior to x. Strong revocation therefore has a cascading effect up-wards in the role hierarchy.

The following algorithm shows the strong revocation of r_j from a user with the role set R by an AD role.

Strong revocation algorithm

```
Strong_revoke(ADrole, $R$, $r_j$)
Input: ADrole, a set of roles $R$ and a role $r_j$.
Output: true, if it can strong revoke role $r_j$ from $R$;
false otherwise.
Begin:
    if $r_j \notin R^*$,
    return false;
    /* there is no effect of the strong revocation since
    the user is not an explicit and implicit member
    of $r_j$ */
    else,
    1. if $r_j \in R$ is a mobile member of the user, do
    Weak_revoke(ADrole, $R$, $r_j$) as $u \in EMr_j$
    /* $r_j$ as a mobile member is weakly revoked */;

    2. if $r_j \in R$ is a immobile member of the user, do
    Weak_revoke(ADrole, $R$, $r_j$) as $u \in EIMr_j$;
    /*$r_j$ is weakly revoked from $R$*/;

    3. Suppose
    $Sen = R^* \cap \pi_{Senior}(\sigma_{Junior=r_j}(SEN-JUN))$,
    for all $y \in Sen$ with mobile membership with
    the user, do
    Weak_revoke(ADrole, $R$, $y$) as $u \in EMy$;
    for all $y \in Sen$ with immobile membership with
    the user, do
    Weak_revoke(ADrole, $R$, $y$) as $u \in EIMy$;
    /*the user is weakly revoked from all such user's
    members $y \in Sen$ */.
    If all the weak revocations are successful,
    return true;
    otherwise, return false.                          ◇
```

It should be noted that Weak_revoke(ADrole, R, r_j) and Weak_revoke(ADrole, R, y) do not work if ADrole has no right to revoke r_j. We have the following consequence.

Theorem 3 The explicit and implicit mobile and immobile members of role r_j are revoked from the user by the Strong revocation algorithm $Strong\_revoke(ADrole, R, r_j)$ if the ADrole has the right to revoke r_j from the *Can-revoke-M* and *Can-revoke-IM* relations.

Corollary 1 The authorization revocation problem is solved by the Weak revocation algorithm and Strong revocation algorithm.

In the remaining parts of this paper, the new relational algebra approaches is used with a payment scheme.

4 Applications of the relational algebra algorithms

The new relational algebra algorithms are applied to a consumer anonymity scalable payment scheme (Wang H., Cao J. and Kambayashi Y. 2002) in this section. We first briefly recall the payment scheme and consider the relationships of the roles in the scheme, then analyze applications of the relational algebra algorithms.

4.1 Review of the anonymity scalable electronic payment scheme

The payment scheme in (Wang H., Cao J. and Kambayashi Y. 2002) provides different degrees of anonymity for consumers. Consumers can decide the levels of anonymity. They can have a low level of anonymity if they want to spend coins directly after withdrawing them from the bank. Consumers can achieve a high level of anonymity through an anonymity provider (AP) agent without revealing their private information and are secure in relation to the bank because the new certificate of a coin comes from the AP agent who is not involved in the payment process.

Electronic cash has sparked wide interest among cryptographers ((Rivest R. T. 1992, Yiannis T. 1998, Okamoto T. 1995)). In its simplest form, an e-cash system consists of three parts (a bank, a consumer and a shop) and three main procedures withdrawal, payment and deposit. Besides the basic participants, a third party named anonymity provider (AP) agent is involved in the scheme. The AP agent helps the consumer to get the required anonymity but does not involve in the purchase process. The model is shown in Figure 6. The AP agent gives a certificate to the consumer when s/he needs a high level of anonymity.

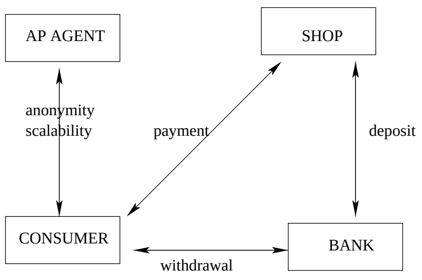
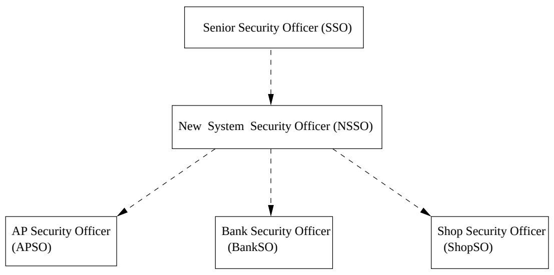


Figure 6: Electronic cash model

From the viewpoint of banks, consumers can improve anonymity if they are worried about disclosure of their identities. This is a practical payment scheme for Internet purchases because it has provided a solution with different anonymity requirements for consumers. However, consumers cannot get the required level of anonymity if the role BANK and AP are assigned to a user. It means the management of the payment scheme is an important issue. To simplify the management of the scheme and prevent errors in the user-role assignment, we analyze its management with the relational algebra algorithms.

4.2 An application of the authorization granting algorithm

We only show an application of the authorization granting algorithm since the length limitation. A hierarchy of roles and a hierarchy of administrative roles are shown in Figure 7 and Figure 8 respectively. Table 9 shows the *Can-assign-M* relations with the prerequisite conditions in the scheme and Table 10 shows ROLES relationship (the attribute M1C, for example, shows whether the role M1 is conflicting with the RoleName in the relation or not). We only show how to assign and revoke a role as a mobile member of a user.



Senior Security Officer (SSO)
New System Security Officer (NSSO)
AP Security Officer (APSO)
Bank Security Officer (BankSO)
Shop Security Officer (ShopSO)

Figure 8: Administrative role assignment in the scheme

Let us consider the APSO tuples in Table 9 (the analysis for BankSO and ShopSO are similar). The first tuple authorizes APSO to assign users with the prerequisite role FPS into mobile members in the AP agent (AP). The second one authorizes APSO to assign users with the prerequisite condition $FPS \wedge \overline{OP}$ to be mobile member of quality controllers (QC). Similarly, the third tuple authorizes APSO to assign users with the prerequisite condition $FPS \wedge \overline{QC}$ to be mobile operators (OP). The second and third tuple show that the APSO can grant a user who is a member of the AP agent into one but not both of QC and OP. This illustrates how mutually exclusive roles can be enforced. However, for the NSSO and SSO these are not mutually exclusive. The fourth tuple authorizes APSO to put a user who is a member of both QC and OP into a mobile manager (M1). Of course a user could have become a member of both QC and OP only by actions of a more powerful administrator than APSO.

Assume Bob is a user with role FPS. The administrative role (NSSO) wants to assign the role AP to Bob. Using the granting algorithm *Grant(NSSO, FPS, AP)*, the first step, $R^* = R = \{FPS\}$ and

$$\pi_{Prereq.Condition}(\sigma_{Admin.role=NSSO}(Can-assign-M)) = \{FPS\},$$

then

$$R^* \cap \pi_{Prereq.Condition}(\sigma_{NSSO}(Can-assign-M)) \neq \phi.$$

This means NSSO can assign role AP as a mobile member to Bob. The second step, based on Table 10,

$$Num = \pi_{APC}(\sigma_{RoleName=FPS}(ROLES)) = 0$$

Admin.role	Prereq.Condition	Role Range
APSO	FPS	{AP}
APSO	FPS $\wedge\overline{OP}$	{QC}
APSO	FPS $\wedge\overline{QC}$	{OP }
APSO	QC $\wedge$ OP	{M1 }
BankSO	FPS	{Bank}
BankSO	FPS $\wedge\overline{TE} \wedge \overline{AU}$	{AC}
BankSO	FPS $\wedge\overline{TE} \wedge \overline{AC}$	{AU}
BankSO	FPS $\wedge\overline{AU} \wedge \overline{AC}$	{TE}
BankSO	TE $\wedge AU \wedge$ AC	{M2 }
ShopSO	FPS	{Shop }
ShopSO	FPS $\wedge\overline{SELLER}$	{AUDITOR}
ShopSO	FPS $\wedge\overline{AUDITOR}$	{SELLER}
ShopSO	SELLER $\wedge$ AUDITOR	{M3}
NSSO	FPS	(FPS, DIR)
SSO	E	[FPS, FPS]
SSO	FPS	(FPS, DIR]

Table 9: Can-assign-M relation in the scheme

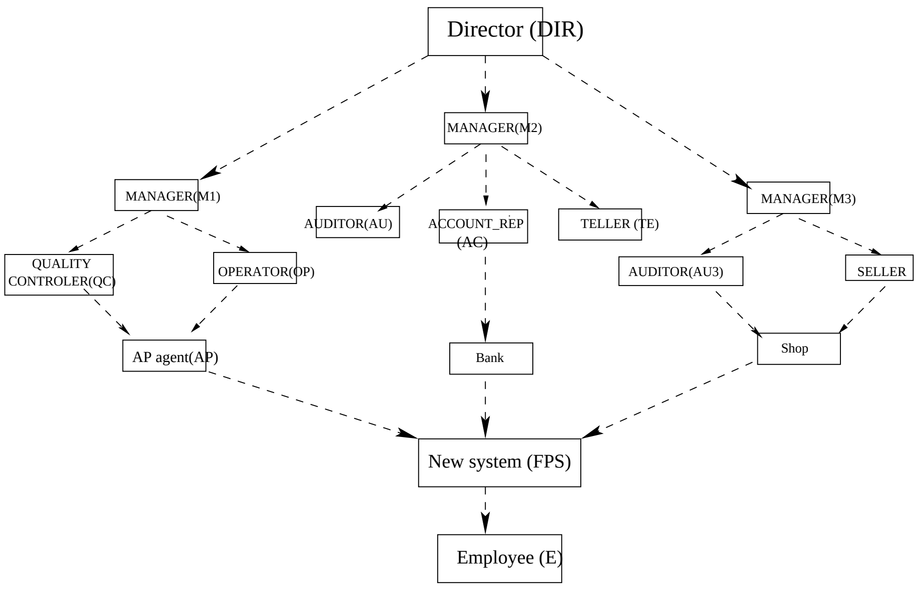
It means no conflicts when assigning AP to Bob.

5 Related work

There are several other related works on role-based access control models with mobility of user-role relationship (Sandhu R. and Munawer Q. 1999), formal authorization approaches for RBAC (Wang H., Cao J. and Zhang Y. 2002) and user-role assignment mechanisms developed for web-based intranets (Sandhu R. and Park J. S. 1998).

An administrative role based access control model for administration of roles was introduced in (Sandhu R. and Munawer Q. 1999). The authors analyzed several subtle issues such as user-role assignment, permission-role assignment with mobilities of user-role membership and permissions-role membership. The conception of mobility also appeared in this paper. However, our work substantially differs from (Sandhu R. and Munawer Q. 1999) in two aspects. First, the paper (Sandhu R. and Munawer Q. 1999) only introduced the definition of mobility of user-role membership in user-role assignment. By contrast, we discuss in detail various cases and focus on possible problems in user-role assignment with mobility of use-role relationship. Second, authors of the paper (Sandhu R. and Munawer Q. 1999) describe the management of user-role assignment with mobility, but they did not mention conflicts when assigning roles to users. Therefore, there is no support to deal administrative roles with regular roles in the proposal, especially mobile and immobile members. By contrast, we presented a number of specialized authorization algorithms for access control which allow administrators to authorize a role as mobile and immobile member to or revoke them from users. These algorithms provide a rich variety of options that can handle the document of administrative roles with regular roles as mobile and immobile members.

In our earlier work (Wang H., Cao J. and Zhang Y. 2002), formal authorization approaches for user-role assignment have been developed. This paper is an extension version of the work in (Wang H., Cao J. and Zhang Y. 2002). If all membership is restricted to being mobile, the algorithms in this paper are identical to the algorithms in (Wang H., Cao J. and Zhang Y. 2002). This can be achieved by setting two relations *Can-assign-IM* and *Can-revoke-IM* to be empty. There are two main different points in these two papers. The first is that we discuss mobile and immobile memberships in this paper but these two memberships were not analyzed in paper (Wang H., Cao J. and Zhang Y. 2002). The second is that there are no prerequisite conditions in the revocation model in



AP agent:

The Manager inherits the Operator and Quality controler. They are employees

Bank:

The Manager inherits the Teller, Auditor and Account_rep, they are employees. The Account_rep has DSD relatonships with the Teller, SSD relationship with the Auditor.

Shop:

The manager inherits the Saler and the Auditor, they are employees. The Saler has DSD relationship with the Auditor.

Figure 7: User-role assignment in the payment scheme

RoleName	EC	FPSC	APC	QCC	OPC	M1C	BaC	TEC	ACC	AUC	M2C	ShC	SAC	AU3C	M3C	DiC
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FPS	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AP	0	0	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0
QC	0	0	0	0	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0
OP	0	0	0	-1	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0
M1	0	0	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0
Bank	0	0	-1	-1	-1	-1	0	0	0	0	0	-1	-1	-1	-1	0
TE	0	0	-1	-1	-1	-1	0	0	0	-1	0	-1	-1	-1	-1	0
AC	0	0	-1	-1	-1	-1	0	0	0	-1	0	-1	-1	-1	-1	0
AU	0	0	-1	-1	-1	-1	0	-1	-1	0	0	-1	-1	-1	-1	0
M2	0	0	-1	-1	-1	-1	0	0	0	0	0	-1	-1	-1	-1	0
Shop	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0
SELLER	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	-1	0	0
AUDITOR	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	0	0	0
M3	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0
Director	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 10: ROLES in the payment scheme

(Wang H., Cao J. and Zhang Y. 2002) but in this paper there are.

Finally, a user-role assignment for web-based intranet has been proposed in (Sandhu R. and Park J. S. 1998). The RBAC/web system with user-role assignment model (URA97) (Sandhu R. and Bhamidipati V. North-Holland, 1997) is extended to decentralize the details of RBAC administration on the web. Authorizations can be given by administrators through prerequisite conditions. However, there are two differences between their work and our work in this paper. Firstly, they did not discuss the mobility of users but we do. Secondly, there is a consistency problem discovered in (Sandhu R. and Park J. S. 1998):

It is very difficult to keep the consistency by reflecting security requirements between global network objects and local network objects if there are hundreds of roles and thousands of users in a system.

We proposed a number of authorization algorithms that can be used to address the consistency problem. The authorization granting algorithm is used to find conflicts and prevent the derivation of certain information while the strong revocation algorithm is used to check whether a role is permitted to have permissions of another role.

6 Conclusions

This paper has discussed the mobility of user-role relationship in RBAC management and provided new authorization allocation algorithms for RBAC along with the mobility that are based on relational algebra operations. They are the authorization granting algorithm, weak revocation algorithm and strong revocation algorithm. The algorithms can automatically check conflicts when granting more than one role as mobile and immobile members to a user in a system. The complexities of the algorithms are also analyzed. Furthermore, we have discussed how to use the algorithms for the electronic payment scheme.

References

Barkley J. F., Beznosov K. and Uppal J. (1999), Supporting relationships in access control using role based access control, *in* 'Third ACM Workshop on RoleBased Access Control', pp. 55–65.

Bertino E., Castano S., Ferrari E. and Mesiti M. (2000), 'Specifying and enforcing access control policies for XML document sources', *World Wide Web, 3* pp. 139–151.

David F. F., Dennis M. G. and Nickilyn L. (1993), An examination of federal and commercial access control policy needs, *in* 'NIST NCSC National Computer Security Conference', Baltimore, MD, pp. 107–116.

Feinstein H. L. (1995), Final report: Nist small business innovative research (sbir) grant: role based access control: phase 1. technical report, *in* 'SETA Corp.'.

Ferraiolo D. F. and Kuhn D. R. (1992), Role based access control, *in* '15th National Computer Security Conference', http://www.citeseer.nj.nec.com/, pp. 554–563.

Ferraiolo D. F., Barkley J. F. and Kuhn D. R. (1999), Role-based access control model and reference implementation within a corporate intranet, *in* 'TISSEC', Vol. 2, pp. 34–64.

Oh S. and Sandhu R. (2002), A model for role administration using organization structure, *in* 'Seventh ACM Symposium on Access Control Models and Technologies', ACM Press, pp. 155–162.

Okamoto T. (1995), An efficient divisible electronic cash scheme, *in* 'Advances in Cryptology–Crypto'95', Vol. 963 of *Lectures Notes in Computer Science*, Springer-Verlag, pp. 438–451.

Rivest R. T. (1992), 'The MD5 message digest algorithm', *Internet RFC 1321* .

Sandhu R. (1998*a*), Role activation hierarchies, *in* 'Third ACM Workshop on RoleBased Access Control', ACM Press, pp. 33–40.

Sandhu R. (1998*b*), 'Role-Based Access Control', *Advances in Computers* **46**.

Sandhu R. and Bhamidipati V. (North-Holland, 1997), 'The ura97 model for role-based administration of user-role assignment', *T. Y. Lin and Xiao Qian, editors, Database Security XI: Status and Prospects* pp. 262–275.

Sandhu R. and Munawer Q. (1999), The arbac99 model for administration of roles, *in* 'the Annual Computer Security Applications Conference', ACM Press, pp. 229–238.

Sandhu R. and Park J. S. (1998), Decentralized User-Role Assignment for Web-based Intranets, *in* '3th ACM Workshop on Role-Based Access Control', ACM Press, pp. 1–12.

Wang H., Cao J. and Kambayashi Y. (2002), Building a consumer anonymity scalable payment protocol for the internet purchases, *in* '12th International Workshop on Research Issues on Data Engineering: Engineering E-Commerce/E-Business Systems', San Jose, USA.

Wang H., Cao J. and Zhang Y. (2002), Formal authorization allocation approaches for role-based access control based on relational algebra operations, *in* '3nd International Conference on Web Information Systems Engineering (WISE02)', Singapore, pp. 301–312.

Wang H., Sun L., Cao J., and Zhang Y. (2004), Anonymous access scheme for electronic-services , *in* 'Proceedings of the Twenty-Seventh Australasian Computer Science Conference (ACSC2004)', Dunedin, New Zealand, pp. 296–305.

Wang H., Zhang Y., Cao J., Kambayahsi Y. (2004), 'A global ticket-based access scheme for mobile users', *Special Issue on Object-Oriented Client/Server Internet Environments, Information Systems Frontiers* **6**(1), 35–46.

Wang H., Zhang Y., Cao J., Varadharajan V. (2003), 'Achieving secure and flexible m-services through tickets', *IEEE Transactions on Systems, Man, and Cybernetics, Part A, Special issue on M-Services* pp. 697–708.

Yiannis T. (1998), Fair off-line cash made easy, *in* 'Advances in Cryptology–Asiacrypt'98', Vol. 1346 of *Lectures Notes in Computer Science*, Springer-Verlag, pp. 240–252.