# An Attribute-Based Delegation Model and Its Extension

**Chunxiao Ye and Zhongfu Wu**

College of Computer Science
Chongqing University
Chongqing, 400044, China
Email:yecx@cqu.edu.cn

**Yunqing Fu**

College of Distance Education
Chongqing University
Chongqing, 400044, China

*In existing delegation models, delegation security entirely depends on delegators and security administrators, for delegation constraint in these models is only a prerequisite condition. This paper proposes an Attribute-Based Delegation Model (ABDM) with an extended delegation constraint consisting of both delegation attribute expression (DAE) and delegation prerequisite condition (CR). In ABDM, a delegatee must satisfy delegation constraint (especially DAE) when assigned to a delegation role. With delegation constraint, a delegator can restrict the delegatee candidates more strictly. ABDM relieves delegators and security administrators of security management work in delegation. In ABDM, a delegator is not allowed to temporarily delegate permissions to a person who does not satisfy the delegation constraint. To guarantee its flexibility and security, an extension of ABDM named $ABDM_X$ is proposed. In $ABDM_X$, a delegator can delegate some high level permissions to low level delegatee candidates temporarily, but not permanently.*

*Keywords: Access Control, RBAC, Delegation, Attribute*
*ACM Classification: D.4.6 (Operation System-Security and Protection-Access Control)*

## 1. INTRODUCTION

Access control is one of the most important security technologies in information systems. As an alternative to DAC and MAC, Role-Based Access Control (RBAC) (Sandhu *et al*, 1996) security technology has gained considerable attention (Ferraiolo *et al*, 2001) recently.

Delegation means a delegator can assign his/her permissions to a delegatee. There are three types of situations in which delegation takes place: backup of roles, decentralization of authority and collaboration of work (Zhang *et al*, 2003). Many studies have been done in delegation (Stein, 1987; Moffett, 1990; Gasser and McDermott, 1990), and considerable attention is paid to human-to-human delegation (Zhang *et al*, 2003; Barka and Sandhu, 2000a; Barka and Sandhu 2000b; Barka , 2002; Barka and Sandhu 2004).

But there are still some problems in delegation needing to be solved:



*Manuscript received: 12 April 2005*
Communicating Editor: Julio Cesar Hernandez

1. Because delegation is controlled by the delegator itself, a malicious user can delegate some important permissions to low level delegatees.
2. The delegation security relies heavily on the security administrator.
3. Delegation prerequisite condition cannot restrict the scope of delegatees more strictly.
4. It is difficult for a delegator to select qualified delegatees.

In this paper we first propose a new delegation model named Attribute-Based Delegation Model (ABDM). Delegation constraint in ABDM consists of both delegation prerequisite condition (CR) and delegation attribute expression (DAE). Only those delegatees whose prerequisite roles and DAE satisfy CR and DAE of delegation constraint can be assigned to a delegation role. In ABDM, DAE and CR form a strict delegation constraint in delegation. ABDM is a strict and secure delegation model both in temporary and permanent delegation.

But sometimes we need a less strict delegation model in temporary delegation, such as high level permissions temporarily delegated to low level users. Since ABDM does not support this kind of delegation, we propose a delegation model named $ABDM_X$ to solve this problem, which is an extension of ABDM.

The rest of this paper is organized as follows. Section 2 presents related work. In Section 3, we introduce ABDM model. Section 4 presents the $ABDM_X$ model. Section 5 is a discussion among ABDM, $ABDM_X$ and some existing delegation models. Conclusions and future works are presented in Section 6.

## 2. RELATED WORKS AND MOTIVATION

### 2.1 Related Works

RBDM (Barka and Sandhu, 2000a; Barka and Sandhu, 2000b; Barka, 2002; Barka and Sandhu 2004) is the first delegation model based on role. In RBDM, a user can delegate his/her role to another user. A rule-based declarative language has been proposed in RDM2000 (Zhang *et al*, 2001) to specify and enforce policies in delegation. The delegation unit in RBDM and RDM2000 is "role". In RPRDM (Zhao *et al*, 2003), a delegator can delegate part of their permissions to a delegatee by a "mask". Tamassia *et al* (2004) proposed a role-based cascaded delegation model in decentralized trust management systems.

PBDM (Zhang *et al*, 2003) is a flexible delegation model that supports multi-step delegation and revocation in role and permission level. In PBDM0, a user can delegate all or part of his/her permissions to delegatees. In PBDM1 and PBDM2, the permission flow is managed by a security administrator with delegeatable role (DBR). RDM2000 and RBDM can be seen as special cases of PBDM.

In most cases, a delegator cannot delegate all of their permissions to delegatees. Therefore, a low level user cannot be assigned to high level permissions. In some delegation models, delegation is managed by the delegator him/herself. RPRDM only addresses repeated and partial delegation, and delegation in RPRDM is also controlled by delegators. So is the delegation in PBDM0. In PBDM1 and PBDM2, delegation is managed by system administrators or organization security administrators, and a delegator cannot delegate high level permissions to low level users.

RDM2000 and PBDM use can-delegate condition with prerequisite condition to restrict delegatees, but the prerequisite condition in these models consists only of prerequisite role or organization unit (Sandhu *et al*, 1999; Sandhu and Munawer, 1999; Sandhu and Munawer, 2002). RBAC and other delegation models overlook the differences among users who have the same roles. They are all on the assumption that users who satisfy the prerequisite condition of a delegation permission can be assigned to the delegation permissions, but in some cases this is not true.

Role and user attribute has been proposed recently (Goh and Baldwin, 1998; Al-Kahtani and Sandhu, 2002; Al-Kahtani, 2003). In RB-RBAC (Al-Kahtani and Sandhu, 2002; Al-Kahtani, 2003), users who have attribute expression will be assigned to roles dynamically and automatically. Attribute expression in Ye *et al* (2004) indicates the user's qualifications and abilities required by a role.

### 2.2 Motivation

The prerequisite condition of existing delegation models only consists of prerequisite role, which indicates the qualifications and abilities of users. In fact, the prerequisite role cannot distinguish one user from the others in many cases. In RBAC, a role generally can be seen as a position in an organization, while a permission can be seen as a work or task. But in a real organization, there are only a few positions and users in the same position may have different qualifications and abilities. One cannot create different roles for different users with different qualifications and abilities for it will increase the total number of roles remarkably.

Figure 1 shows a role hierarchy in a software company. When the Quality Engineer (*QE*) is on a business trip, part of their works must be delegated to someone else, such as a *programmer*. Suppose *QE* wants to delegate their permissions *Inspect-Java-code*, *Inspect-VB-code* and *Inspect-Delphi-code* to programmers. Obviously, these permissions have some requirements on users' qualifications and abilities, for a user without the required qualifications and abilities can hardly fulfill the work. For example, *Inspect-Java-code* requires the user has the qualifications and abilities of being a JAVA coder for at least two years. In existing delegation models, the prerequisite condition of this example is only a role *programmer*. For example, in RBDM, RDM2000 and PBDM, the delegation can be restricted by *can-delegate* (*QE, programmer*), *can-delegate* (*QE, programmer, n*) and *can-delegate* (*QE, programmer,* {*Inspect-Java-code*}, *n*) respectively, where the delegation prerequisite conditions are the same: *programmer*.

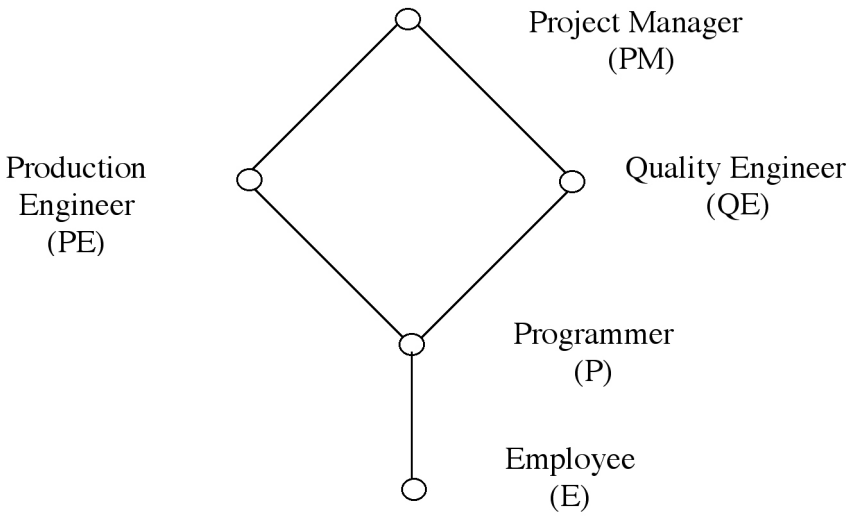



**Figure 1: Example of role hierarchy**

| User | Qualifications and abilities | Roles |
|---|---|---|
| Alex | Three years Java programming experience | Programmer, employee |
| Annie | Two years VB programming experience | Programmer, employee |
| Betty | One year Java programming experience | Programmer, employee |
| John | Two years Java programming experience | Programmer, employee |
| Lucy | Two years Delphi programming experience | Programmer, employee |
| Mary | Three years VB programming experience | Programmer, employee |
| Mike | Five years Java programming experience | Programmer, employee |
| Tony | Two years Delphi programming experience | Programmer, employee |

**Table 1: Users' qualifications, abilities and roles**

In Table 1, users with different qualifications and abilities can have the same role *programmer*. So, it is difficult for the *QE* to select qualified delegatees for they cannot distinguish one user from the others only by the role *programmer*. Sometimes it will cause an unsafe delegation. For example, *QE* may delegate the permission *Inspect-VB-code* to a JAVA coder or the permission *Inspect-Java-code* to a JAVA coder who has only one year's experience. To ensure a safe delegation, *QE* must select a user who has the required qualifications and abilities and delegate permissions to him/her. But in a situation of numerous users, this will burden the delegator or security administrator's work.

To solve this problem, we introduce the concept of user attribute with which to describe the difference among users and ensure a convenient and secured delegation.

## 3. ABDM MODEL

Delegations in ABDM are divided into two types: decided-delegatees and undecided-delegatees. For example, when a finance manager (*FM*) is out of work, part of the *FM*'s permissions can be delegated to a person, say *Tom*, if *Tom* has the required qualifications and abilities. This is a decided-delegatee delegation. In another case, the *FM* may want to delegate some permissions to a user who has the required qualifications and abilities, but does not know who has the required qualifications and abilities. If the system can generate qualified delegatee candidates automatically, the *FM* can choose one of the candidates as a delegatee. This is an undecided-delegatee delegation. ABDM can solve these problems mentioned in Section 1 and make delegation securer and easier by decided-delegatee and undecided-delegatee delegation.

The delegation in ABDM is similar to that in PBDM. In ABDM, a delegator must first create a temporary delegation role, say *tdr*, and then assigns permissions to *tdr*. Finally, they can assign users to *tdr*. In delegation, the temporary delegation role has the same function as that of *DTR* in PBDM. With temporary delegation role, ABDM supports partial delegation. Unlike PBDM1 and PBDM2, there is no *DBR* in ABDM, for its function in delegation can be replaced by a temporary delegation role.

The delegation constraint in our delegation model consists of both prerequisite condition (CR) (Zhang *et al*, 2001) and delegation attribute expression (DAE). Only the persons who satisfy both CR and DAE can be assigned to a temporary delegation role. Users with different DAEs can be assigned to different delegation roles temporarily. With DAE and CR, ABDM has a stricter constraint in delegation.

### 3.1 Concepts

**Definition 1**

DAE::= uae {AND uae}
uae::= ua roprt uav
roprt::= '<'|'≤'|'='|'≥'|'>'|'≠'
ua::= {*specified by system*}
uav::= {*specified by system*}

, where DAE is delegation attribute expression associated with permission. uae, roprt, ua and uav are attribute expression, attribute relation expression operator, attribute name, attribute value respectively. AND is the usual logic operator 'and'.

For example, *level*=4, *type*='*S*' and *total*≥33 are uaes or DAEs, while *total*≥20 AND *type*='*S*' and *total*≥20 AND *type*='S' AND *total*≥33 are DAEs.

In some of the existing models (Al-Kahtani and Sandhu, 2002; Al-Kahtani, 2003), only users can have attribute expression. The substantial improvement on it made by our work is that both users and permissions in ABDM have DAEs. A user's DAE indicates the user's qualifications and abilities, while a permission's DAE indicates a delegatee's qualifications and abilities required by the permission in delegation.

For convenience of understanding, we use *u*.DAE, *p*.DAE and *tdr*.DAE to denote the DAE of a user *u*, a permission *p* and a temporary delegation role *tdr* respectively.

**Definition 2** $uae_i$ and $uae_j$ are said to have identical structures if and only if they have the same uas and roprts. $uae_i$ and $uae_j$ are said to be comparable if they have identical structures, otherwise they are incomparable.

For example, *level*=4 and *level*=5 are comparable, while *level*=4 and *level*≥5 are incomparable.

Similar to recent studies (Al-Kahtani and Sandhu, 2002; Al-Kahtani, 2003), we use the symbol '≥' to denote the dominance relations between two comparable uaes. We can use those rules shown in Table 2 to determine the dominance relation between two comparable uaes.

| Rule | $uae_1$ | $uae_2$ | Dominance relation |
|---|---|---|---|
| Rule1 | $ua \geqslant uav_1$ or $ua > uav_1$ | $ua \geqslant uav_2$ or $ua > uav_2$ | $uae_1 \geq uae_2$ if and only if $uav_1$ and $uav_2$ are numeric or date value and $uav_1 \geq uav_2$; for other data type, the dominance relation must be specified by system. |
| Rule2 | $ua \leqslant uav_1$ or $ua < uav_1$ | $ua \leqslant uav_2$ or ua<uav2 | $uae_1 \geq uae_2$ if and only if $uav_1$ and $uav_2$ are numeric or date value and $uav_1 \leq uav_2$; for other data type, the dominance relation must be specified by system. |
| Rule3 | $ua = uav_1$ | $ua = uav_2$ | $uae_1 \geq uae_2$ if and only if $uav_1 = uav_2$. |
| Rule4 | $ua \neq uav_1$ | $ua \neq uav_2$ | The dominance relation must be specified by system. |

**Table 2: Dominance relations among uaes**

For example, we can say $uae_1$ (*level*>5) ≥ $uae_2$ (*level*>4) and $uae_3$ (*total*≤20) ≥ $uae_4$ (*total*≤30). The dominance relations between $uae_5$ (*type*≠'*S*') and $uae_6$ (*type*≠'*J*') must be manually specified.

We can say $uae_i$ dominates $uae_j$ if $uae_i \geq uae_j$. In this case, $uae_i$ is the dominant uae and $uae_j$ is the non-dominant one.

A temporary delegation role *tdr* has its own DAE, which is a combination of DAEs of its permissions. *tdr*.DAE can be automatically generated by the system. Permissions' DAEs will only be used to generate a temporary delegation role's DAE in delegation. So, dominance relation can only be tested between a user's DAE and a temporary delegation role's DAE.

For convenience of understanding, we use UAE to denote the uae set of a DAE. For example, the UAE of *level*>5 *AND total*≤20 is {*level*>5, *total*≤20}.

We use '≥' to denote the dominance relation between two DAEs:

**Definition 3** We say $DAE_1 \geq DAE_2$, if $\forall$ $uae_j \in UAE_2$, $\exists$ $uae_i \in UAE_1$, s.t. $uae_i \geq uae_j$, where $UAE_1$ and $UAE_2$ are uae sets of $DAE_1$ and $DAE_2$ respectively.

In this case, $DAE_1$ is the dominant DAE and $DAE_2$ is the non-dominant one.

For example, we can say $DAE_1$ (*level*>5 *AND total*≤20) ≥ $DAE_2$ (*level*>4 *AND total*≤30) for *level*>5 ≥ *level*>4 and *total*≤20 ≥ total≤30. We can also say $DAE_3$ (*level*>5 *AND total*≤20) ≥ $DAE_4$ (*level*>4) according to definition 3.

We say a user is a qualified delegatee of *tdr* if their DAE ≥ *tdr*.DAE in delegation, otherwise they are an unqualified delegatee of *tdr*.

Here we introduce a DAE generation algorithm named DG algorithm as below:

**DG** (DAE Generation) Algorithm:

```
Input: p1…pn∈P, where P is the permission set of tdr.
Output: DAE of tdr
Begin
  UAE=ϕ;
  for i=1 to n
    UAE=UAE∪UAEi, where UAEi is the uae set of pi.DAE
  for i=1 to |UAE|
    for j=1 to |UAE|
    if uaei≠uaej and uaei ≥ uaej then delete uaej from UAE
  Return DAE =uae1 AND…AND uaen, where uae1…uaen ∈UAE, n=|UAE|
End
```

In the DG algorithm, comparable uaes are tested for dominance relation one by one, and the non-dominant ones are discarded. In the end, only incomparable uaes remain in UAE and these uaes can form the *tdr*.DAE. Each uae in *tdr*.DAE has its own restriction on user's corresponding uae. Because uaes in *tdr*.DAE have the strictest restrictions on users, a delegator cannot delegate high level permissions to unqualified users. So, *tdr*.DAE generated by the DG algorithm can reflect the comprehensive requirements of users' DAEs required by delegation permissions and thus guarantee the security of delegation.

For example, suppose $p_1$.DAE is *level*>5 AND *total*≤40 and $p_2$.DAE is *level*>4 AND *total*≤30. The output of **DG** ($p_1$, $p_2$) is *level*>5 AND *total*≤30 because *level*>5 ≥ *level*>4 and *total*≤30 ≥ *total*≤40.

### 3.2 ABDM

**Definition 4** The following is a list of ABDM components:

- *R, RR, TDR, S, P, U, Ude*, and *Uee* are set of roles, regular roles, temporary delegation roles, sessions, permissions, users, decided-delegatee candidates and undecided-delegatee candidates respectively.
- $RH \subseteq RR \times RR$ is a regular role hierarchy.

- $TDRH_u \subseteq TDR \times TDR$ is a temporary delegation role hierarchy owned by a user $u$.
- $R=RR \cup TDR$
- $RR \cap TDR=\phi$
- $URA \subseteq U \times RR$ is a user to regular role assignment relation.
- $UDA \subseteq Ude \times TDR$ is a decided-delegatee to temporary delegation role assignment relation.
- $UEA \subseteq Uee \times TDR$ is an undecided-delegatee to temporary delegation role assignment relation.
- $UA=URA \cup UDA \cup UEA$
- $PRA \subseteq P \times RR$ is a permission to regular role assignment relation.
- $PDA \subseteq P \times TDR$ is a permission to temporary delegation role assignment relation.
- roles: $U \rightarrow 2^R$ is a function mapping a user to a set of roles.
- $roles\ (u) = \{r | (u, r) \in UA\}$
- $per\_r$: $RR \rightarrow 2^P$ is a function mapping a regular role to a set of permissions.
  $per\_r(r) = \{p | (\exists\ r' \leq r)\ (p, r') \in PRA\}$
- $per\_d$: $TDR \rightarrow 2^P$ is a function mapping a temporary delegation role to a set of permissions.
  $per\_d(tdr)=\{p | (\exists\ tdr' \leq tdr)((p, tdr') \in PDA)\}$
- $per\_u$: $U \rightarrow 2^P$ is a function mapping a user to a set of permissions.
  $per\_u(u)=\{p | (\exists\ r \in RR)((u,r) \in URA\ \Lambda(p, r) \in PRA)\} \cup \{p | (\exists\ r \in TDR)((u,r) \in UDA\ \Lambda(p, r) \in PDA)\} \cup \{p | (\exists\ r \in TDR)((u,r) \in UEA\ \Lambda(p, r) \in PDA)\}$
- $Ude$: $TDR \rightarrow 2^U$ is a function mapping a temporary delegation role to a set of users who are assigned to the role.
  $Ude(tdr)= \{u | (\forall p \in per\_d(tdr))(\ p \notin per\_u(u))\ \Lambda\ (u,tdr) \in UDA\}$
- $Uee$: $TDR \rightarrow 2^U$ is a function mapping a temporary delegation role to a set of qualified users.
  $Uee(tdr)=\{u | u.\mathrm{DAE} \geq tdr.\mathrm{DAE}\ \Lambda\ (\forall\ p \in per\_d(tdr))(\ p \notin per\_u(u))\}$
- $can\text{-}delegateD \subseteq R \times \mathrm{CR} \times \mathrm{DAE} \times TDR$ is a delegation constraint on $UDA$.
- $can\text{-}delegateU \subseteq R \times \mathrm{CR} \times Uee \times TDR$ is a delegation constraint on $UEA$.

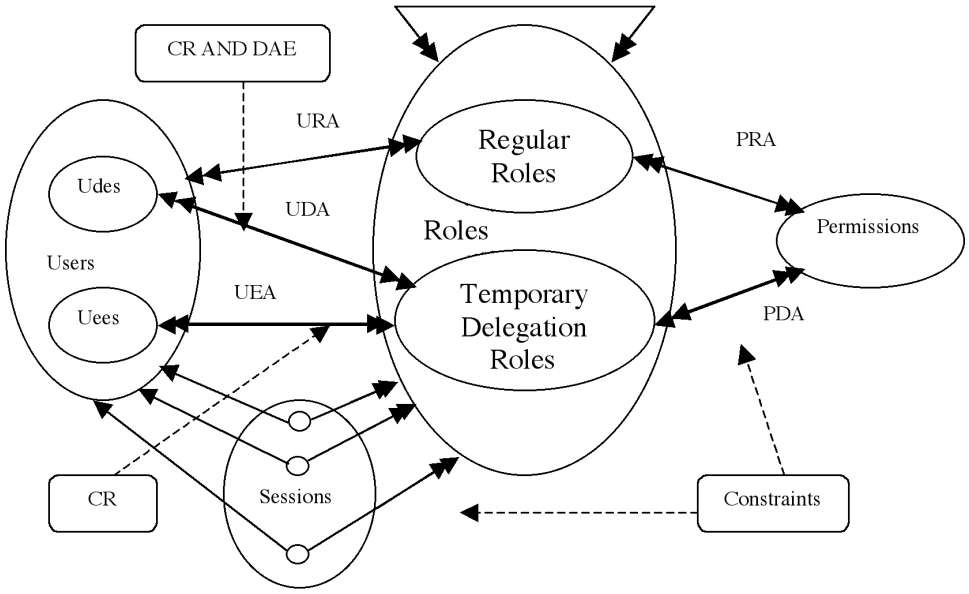



**Figure 2: ABDM model**

User's DAE and Roles

| User | DAE | Roles |
|---|---|---|
| Alex | Language=Java AND years≥3 | P, E |
| Annie | Language=VB AND years≥2 | P, E |
| Betty | Language=Java AND years≥1 | P, E |
| John | Language=Java AND years≥2 | P, E |
| Lucy | Language=Delphi AND years≥2 | P, E |
| Mary | Language=VB AND years≥3 | P, E |
| Mike | Language=Java AND years≥5 | P, E |
| Tony | Language=Delphi AND years≥2 | P, E |

Permission's DAE

| Permission | DAE |
|---|---|
| Inspect-Java-code | Language=Java AND years≥2 |
| Inspect-VB-code | Language=VB AND years≥2 |
| Inspect-Delphi-code | Language=Delphi AND years≥2 |

Role Hierarchy

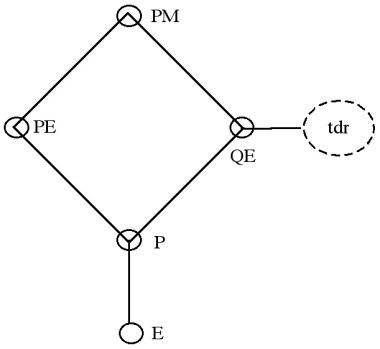



**Figure 3: Example of ABDM**

For example, *can-delegateD* {*ST*, *TR*, *level*=4 AND *type*='*S*' AND *total*=35, *tdr*} means that a delegator who has *ST* can assign a delegatee who must have role *TR* and their DAE satisfies *level*=4 AND *type*='*S*' AND *total*=35 to *tdr*. *can-delegateU*{*ST*, *TR*, *Alex*, *tdr*} means that a delegator who has role *ST* can assign *Alex* to *tdr* if *Alex* is a member of the qualified delegatees set of *tdr* and *alex* has role *TR*.

Here some examples are given to show how ABDM works. Let us discuss the case in Figure 3. For convenience of understanding, we suppose delegatees do not have the same permissions as those of *tdr* before delegation. Figure 3 also gives an example of role hierarchy, user's DAE and its roles, and permission's DAE. *Tom* with a role *QE* is supposed to delegate his permission {*Inspect-Java-code*} to someone. First, he must create a temporary delegation role *tdr*. Second, he can assign permissions {*Inspect-Java-code*} to *tdr*. The *tdr*'s DAE now is "*Language*=*Java* AND *years* ⩾2".

In ABDM, the system can automatically generate a *Uee*(*tdr*) of qualified delegatee candidates after the second step. *Tom* can perform either decided-delegatee or undecided-delegatee delegation.

*Tom* can perform a decided-delegatee delegation according to the following steps:

1. *Tom* selects *Annie* and *Lucy* from user set;
2. *Tom* assigns *Annie* and *Lucy* to *tdr*. The delegation failed for neither *Annie* nor *Lucy* is a qualified delegatee of *tdr*.

*Tom* can perform an undecided-delegatee delegation according to the following steps:

1. *Tom* selects a user, *Alex*, from *Uee*(*tdr*) which is generated by system. In this case, *Uee*(*tdr*)= {*Alex*, *John*, *Mike*}.
2. *Tom* assigns *Alex* to *tdr*. The delegation is successful if *Alex* has role *TR*, otherwise it failed.

Delegation revocation in ABDM is similar to that in PBDM. We believe that delegation revocation with DAE is an interesting topic for further study.

## 4. $ABDM_X$ MODEL

Although a securer delegation model, ABDM still has its shortcoming:

There are two types of delegations: temporary and permanent. ABDM is a delegation model dealing with both types of delegations, and the delegation constraint of a temporary delegation role in these two delegations is identical. But in a real situation, delegation constraint of a temporary delegation role in a temporary delegation is always less strict than that in a permanent delegation. So, with permanent delegation constraint, a delegator sometimes cannot temporarily delegate their permissions to a delegatee.

In the case in Table 3, *p*1 means a teacher who always gives back books without delay can borrow books from the teacher's reading room, and *p*2 and *p*3 means a teacher who has taught a course at least one time can create, administer, grade and record the exam. Suppose a teacher *t* (t has the role *teacher*) requires a student, say *s*, to borrow books from the teacher's reading room on their behalf. They must first create a temporary delegation role *tdr*, and then assign *p*1 to it. Now, *tdr*.DAE is *type*='*T*' AND *without-delay*='*Y*'. In this case, suppose DAE: *type*='*T*'≥DAE: *type*='*S*' and all students have the same DAE: *type*='*S*' but they have not the attribute *without-delay*. *t* cannot perform a decided-delegatee delegation, for *s* is not a qualified delegatee of *tdr*. Then *t* tries to perform an undecided-delegatee delegation. Because none of the students satisfy *tdr*'s DAE, he/she cannot delegate *p*1 to a student in an undecided-delegatee delegation either.

| permissions of teacher | permission's DAE |
|---|---|
| *p*1:borrow books from the teacher's reading room | *type*='*T*' AND *without-delay*='*Y*' |
| *p*2:prepare and administer exam | *type*='*T*' AND *number-of-times*⩾1 |
| *p*3:grade and record exam | *type*='*T*' AND *number-of-times*⩾1 |

**Table 3: Permissions and permissions' DAE**

In fact, there are some differences between *p*1 and *p*2, *p*3: *p*1 can be temporarily delegated to a person who has not the required qualifications and abilities. Actually, in a temporary delegation the DAE of *p*1 is a restriction on the delegators not the delegatees, while in a permanent delegation it is a restriction on the delegatees not the delegators. It will not cause any security problem if *p*1 has been delegated to an unqualified person temporarily. But *p*1 cannot be permanently delegated to an unqualified person, for that will go against security policy. *p*2 and *p*3 can only be delegated to a person if he/she has the required qualifications and abilities both in a temporary and permanent delegation. So, a person with a role *teacher* can delegate their permissions *p*1 to a person temporarily but they cannot temporarily delegate permissions *p*2 and *p*3 to a person in any cases. That is, *p*1 can be delegated to a low level person temporarily but not permanently.

### 4.1 $ABDM_X$

To overcome this shortcoming, we introduce a model named $ABDM_X$, which is an extension of ABDM. In this model, there are two different types of permissions: monotonous permission (*MP*) and non-monotonous permission (*NMP*). *MP* can be temporarily or permanently delegated to a qualified person, while *NMP* can only be temporarily delegated to a low level delegatee candidate.

**Definition 5** a permission *p* is a *MP* if it has an identical restriction on delegatee's DAE both in a temporary and a permanent delegation. *p* is a *NMP* if it has restriction on delegatee's DAE only in a permanent delegation. *MN* (*p*) is a function defined as follows:

$$MN(p) = \begin{cases} True & p \text{ is a } MP \\ False & p \text{ is a } NMP \end{cases}$$

A *NMP* means it has no restriction on delegatee's DAE in a temporary delegation. Permission's monotony must be specified by the system administrator or security administrator beforehand.

**Definition 6** a temporary delegation role *tdr* is a monotonous temporary delegation role if it has an identical restriction on delegatee's DAE both in a temporary and a permanent delegation. *tdr* is a non-monotonous temporary delegation role if it has restriction on delegatee's DAE only in a permanent delegation. *MN* (*tdr*) is a function defined as follows:

$$MN(tdr) = \begin{cases} True & \exists p \in per\_d(tdr), \\ & MN(p) = True \\ False & \forall p \in per\_d(tdr), \\ & MN(p) = False \end{cases}$$

That is, a non-monotonous role has no restriction on delegatee's DAE in a temporary delegation.

Because ABDM does not support delegation with *NMP*s, we must modify it to meet this requirement. To distinguish the different types of delegation, we use symbols PMN and TMP to denote a permanent and a temporary delegation respectively.

**Definition 7** the following is a list of $ABDM_X$ components:

- $R, RR, TDR, S, P, P_M, P_N, U, Ude, Uee, TDR_M, TDR_N$ and $TDR$ are sets of roles, regular roles, temporary delegation roles, sessions, permissions, *MP*s, *NMP*s, users, decided-delegatee candidates, undecided-delegatee candidates, monotonous temporary delegation roles, non-monotonous temporary delegation roles and temporary delegation roles respectively.
- $RH \subseteq RR \times RR$ is a regular role hierarchy.
- $TDRH_u \subseteq TDR \times TDR$ is a temporary delegation role hierarchy owned by a user $u$.
- $TDR = TDR_M \cup TDR_N$
- $TDR_M \cap TDR_N = \phi$
- $R = RR \cup TDR$
- $RR \cap TDR = \phi$
- $P = P_M \cup P_N$
- $P_M \cap P_N = \phi$
- $URA \subseteq U \times RR$ is a user to regular role assignment.
- $UDAM \subseteq Ude \times TDR_M$ is a decided-delegatee to monotonous temporary delegation role assignment.
- $UDAN \subseteq Ude \times TDR_N$ is a decided-delegatee to non-monotonous temporary delegation role assignment.
- $UEAM \subseteq Uee \times TDR_M$ is an undecided-delegatee to monotonous temporary delegation role assignment.
- $UEAN \subseteq Uee \times TDR_N$ is an undecided-delegatee to non-monotonous temporary delegation role assignment.
- $UEA = UEAM \cup UEAN$
- $UDA = UDAM \cup UDAN$
- $UA = URA \cup UDA \cup UEA$
- $PRA \subseteq P \times RR$ is a permission to regular role assignment.
- $PDA \subseteq P \times TDR$ is a permission to temporary delegation role assignment.

- *roles*: $U \rightarrow 2^R$ is a function mapping a user to a set of roles.
  *roles* $(u) = \{r \mid (u,r) \in UA\}$
- *per_r*: $RR \rightarrow 2^P$ is a function mapping a regular role to a set of permissions.
  $per\_r(r) = \{p \mid (\exists\ r' \leq r)(p, r') \in PRA\}$
- *per_d*: $TDR \rightarrow 2^P$ is a function mapping a temporary delegation role to a set of permissions.
  $per\_d(tdr) = \{p \mid (\exists\ tdr' \leq tdr)((p, tdr') \in PDA)\}$
- *per_u*: $U \rightarrow 2^P$ is a function mapping a user to a set of permissions.
  $per\_u(u) = \{p \mid (\exists\ r \in RR)((u,r) \in URA\ \Lambda (p, r) \in PRA)\} \cup \{p \mid (\exists\ r \in TDR)((u,r) \in UDA\ \Lambda (p, r) \in PDA)\} \cup \{p \mid (\exists\ r \in TDR)((u,r) \in UEA\ \Lambda\ (p, r) \in PDA)\}$
- *Ude*: $TDR \rightarrow 2^U$ is a function mapping a temporary delegation role to a set of users that assigned to this role.
  $Ude(tdr) = \{u \mid (\forall\ p \in per\_d(tdr))(p \notin per\_u(u)) \Lambda (u,tdr) \in UDA\}$
- *Uee*: $TDR \rightarrow 2^U$ is a function mapping a temporary delegation role to a set of qualified users.
  $Uee(tdr) = \{u \mid u.DAE \geq tdr.DAE\ \Lambda\ (\forall\ p \in per\_d(tdr))\ (p \notin per\_u(u))\}$
- *Delegation_type* $(u, tdr) = \{\text{PMN} \mid (u, tdr) \in UDA\ \Lambda\ tdr$ has been permanently delegated to $u\}$
- *Delegation_type* $(u, tdr) = \{\text{TMP} \mid (u, tdr) \in UDA\ \Lambda\ tdr$ has been temporarily delegated to $u\}$
- *can-delegateM* $\subseteq R \times \text{CR} \times \text{DAE} \times TDR_M$ is a constraint on *UDAM* in a permanent or temporary delegation.
- *can-delegateTN* $\subseteq R \times \text{CR} \times TDR_N \times \text{TMP}$ is a constraint on *UDAN* in a temporary delegation.
- *can-delegatePN* $\subseteq R \times \text{CR} \times \text{DAE} \times TDR_N \times \text{PMN}$ is a constraint on *UDAN* in a permanent delegation.

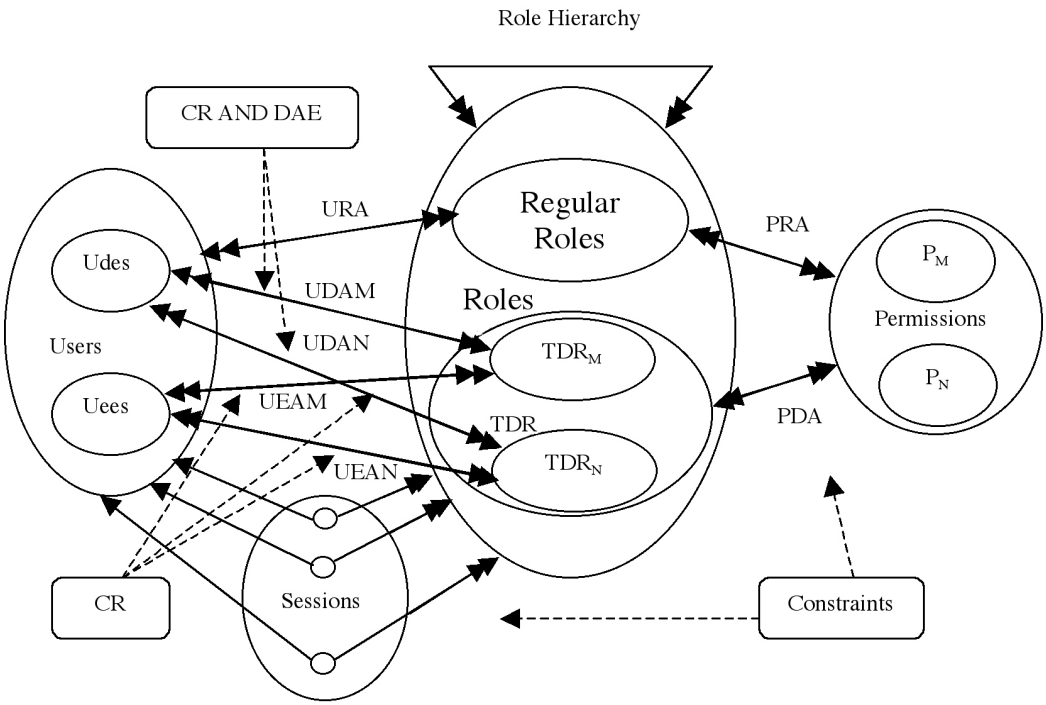



**Figure 4: $ABDM_X$ model**

- *can-delegateMU* $\subseteq R \times$ CR $\times$ *Uee* $\times$ $TDR_M$ is a delegation constraint on *UEAM* in a permanent or temporary delegation.
- *can-delegatePU* $\subseteq R \times$ CR $\times$ *Uee* $\times$ $TDR_N \times$ PMN is a delegation constraint on *UEAN* in a permanent delegation.

With temporary delegation roles' DAEs, *can-delegateM* can restrict delegatees in both types of delegations. *can-delegateTN* can only be used in a temporary delegation with non-monotonous temporary delegation roles, while in a permanent delegation, *can-delegatePN* can restrict delegatees with temporary delegation role's DAE. *can-delegateMU* means a delegator can perform an undecided-delegatee permanent or temporary delegation with monotonous temporary delegation roles, while *can-delegatePU* means a delegator can permanent delegate non-monotonous temporary delegation roles to undecided-delegatees.

Let us discuss the example presented in Section 4 again to show how this extended model works. In one case, teacher *t* wants to temporarily delegate permissions *p*1 to a student *s*. They can delegate it according to the following steps (in Table 3, *p*2, *p*3 are *MP*s and *p*1 is a *NMP*):

1. *t* creates a temporary delegation role *tdr*.
2. *t* assigns *p*1 to *tdr*. That is, *MN* (*tdr*) =*False* for *MN* (*p*1) =*False*.
3. *t* must perform delegation by *UDAN* for $tdr \in TDR_N$.
4. The delegation is successful for *UDAN* (*s*, *tdr*) satisfies *can-delegateTN*(*teacher*, *student*, *tdr*, TMP) constraint.

In the other case, *t* wants to temporarily delegate his/her permissions p1, p2 to *s*:

1. *t* creates a temporary delegation role *tdr*.
2. *t* assigns *p*1, *p*2 to *tdr*.
   That is, *MN* (*tdr*) =*True* for *MN* (*p*1) =*False* and *MN* (*p*2) =*True*, and *tdr*'s DAE is *type*='*T*' AND *without-delay*='*Y*' AND *number-of-times*$\geqslant$1.
3. The delegator must perform delegation by *UDAM* for $tdr \in TDR_M$.
4. The delegation failed because *UDAM*(*s*, *tdr*) does not satisfy *can-delegateM* (*teacher*, *student*, *type*='*T*' AND *without-delay*='*Y*' AND *number-of-times*$\geqslant$1, *tdr*) constraint.

Undecided-delegatee delegation with *MP*s and delegation evocation in $ABDM_X$ are similar to those in ABDM. Revocation in $ABDM_X$ is similar to that in ABDM.

### 4.2 Delegation security in $ABDM_X$

We now discuss delegation security in $ABDM_X$ according to a temporary delegation role *tdr*'s monotony

1. *MN* (*tdr*) =*True*
   In this case, *tdr* has *MP*s and the delegator can perform delegation by *UDAM* or *UEAM*. Because there are not any restrictions on delegation's type in both *can-delegateM* and *can-delegateMU*, a delegatee must be a qualified one when assigned to *tdr* either in a permanent or temporary delegation.
   As we can see in Definition 7, a delegator cannot temporarily or permanently delegate *MP*s to unqualified delegatees.
2. *MN* (*tdr*) =*False*
   In this case, *tdr* has *NMP*s and the delegator can perform delegation with *UDAN* or *UEAN*. Although *can-delegateTN* cannot restrict delegatees with DAE, there will be no security problem in the delegation. The reason is that in fact *NMP*s have no restrictions on delegatees' DAEs in a temporary delegation.

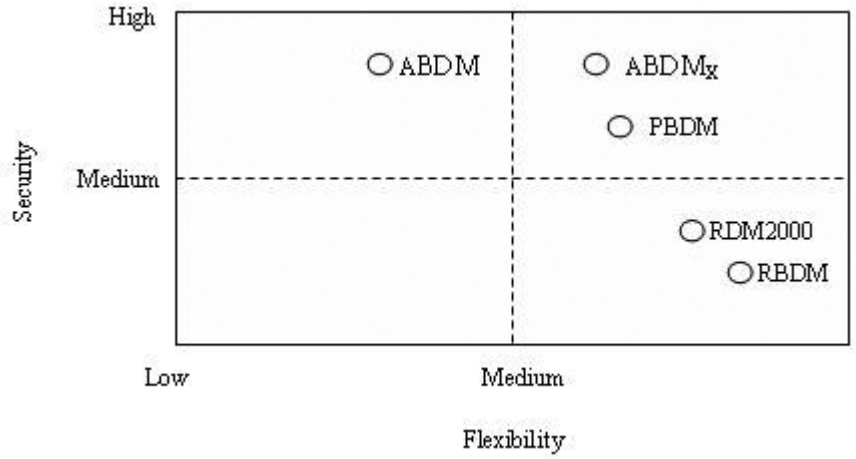



**Figure 5: Security and Flexibility of ABDM, $ABDM_X$, PBDM, RDM2000 AND RBDM**

A delegator cannot permanently delegate *NMP*s to unqualified delegatees by *UDAN* or *UEAN*. In $ABDM_X$, *can-delegatePN* can restrict delegatees with both DAE and CR. That means a delegatee must satisfy both DAE and CR when assigned to a temporary delegation role *tdr* in a permanent delegation. Although *can-delegatePU* cannot restrict delegatees with DAE directly, but the definition of *Uee*(*tdr*) means all members in *Uee*(*tdr*) must satisfy *tdr*.DAE. So a delegator cannot permanently delegate *NMP*s to an unqualified delegatee in an undecided-delegatee delegation.

## 5. DISCUSSION

In some existing delegation models, such as RBDM and RDM2000, delegation is controlled by a delegator or a system administrator. There are no restrictions on delegatee candidates except prerequisite roles. These models have the highest flexibility but lowest security in delegation. In PBDM, a delegator cannot delegate some high level permissions to low level delegatees under the supervision of the system administrator. PBDM has a medium flexibility and security in delegation. ABDM has a strict delegation constraint consisting of prerequisite roles (CR) and temporary delegation role's attribute expression (DAE). A delegatee's prerequisite roles and DAE must satisfy CR and DAE of delegation constraint simultaneously when they are assigned to a temporary delegation role. A delegator cannot delegate high level permissions to an unqualified user in any case. Because delegatee candidates are limited by delegation constraint, ABDM is believed to have the lowest flexibility but highest security in delegation. In $ABDM_X$, a delegator can temporarily delegate *NMP*s to an unqualified low level user but cannot temporarily delegate *MP*s or *NMP*s to an unqualified delegatee in any case. $ABDM_X$ does not cause any security problems in temporary delegation for *NMP*s in fact have no restrictions on delegatee candidates' DAEs. So, $ABDM_X$ has a medium flexibility but the same security level as that of ABDM in delegation.

## 6. CONCLUSION AND FUTURE WORK

We propose a novel delegation model ABDM and its extension $ABDM_X$. As a delegation model based on permission and user's attribute, the main feature of it is that it uses user and permission attribute expression as a part of delegation constraint. ABDM is a securer delegation model for it can restrict delegatee candidates more strictly. $ABDM_X$ is more flexible than ABDM in delegation.

For in $ABDM_X$, a delegator can temporarily delegate *NMP*s to low level users without causing any security problems. Both ABDM and $ABDM_X$ can be used in temporary and permanent delegation and make delegation securer and more flexible.

Further work includes supporting more constraints in ABDM and $ABDM_X$, such as separation of duty and cardinality, and revocation with DAE in them.

## 7. ACKNOWLEDGEMENTS

Our work is supported by The Research Fund for the Doctoral Program of Higher Education (RFDP20040611002), China.

## REFERENCES

AL-KAHTANI, M.A. (2003): A family of models for rule-based user-role assignment. PhD Thesis. School of Information Technology and Engineering, George Mason University.

AL-KAHTANI, M. A. and SANDHU, R. (2002): A model for attribute-based user-role assignment. *Proceedings of the 18th Annual Computer Security Applications Conference*. San Diego California USA, IEEE Computer Society.

BARKA, E.S. (2002): Framework for role-based delegation models. PhD Thesis. School of Information Technology and Engineering, George Mason University.

BARKA, E. and SANDHU, R. (2000a): Framework for role-based delegation models. *Proceedings of 16th Annual Computer Security Application Conference (ACSAC2000)*. New Orleans, USA, IEEE Computer Society.

BARKA, E. and SANDHU, R. (2000b): A role-based delegation model and some extensions. *Proceedings of 23rd National Information Systems Security Conference (NISSC)*. Baltimore, USA, NIST.

BARKA, E. and SANDHU, R. (2004): Role-based delegation model/hierarchical roles (RBDM1). *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*. Tucson, Arizona,USA, IEEE Computer Society.

FERRAIOLO, D.F., SANDHU, R., GAVRILA, S., KUHN, D.R. and CHANDRAMOULI, R. (2001): Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)* 4(3):224–274.

GASSER, M. and McDERMOTT, E. (1990): An architecture for practical delegation in a distributed system. *Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy*. Oakland, USA, IEEE Computer Society.

GOH, C. and BALDWIN, A. (1998): Towards a more complete model of role. *Proceedings of the third ACM workshop on Role-based access control*. Fairfax, Virginia, United States, ACM press.

MOFFETT, J.D. (1990): Delegation of authority using domain based access Rules. PhD Thesis. Dept of Computing, Imperial College, University of London.

SANDHU, R., BHAMIDIPATI, V. and MUNAWER, Q. (1999): The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security* 2(1):105–135.

SANDHU, R. and MUNAWER, Q. (1999): The ARBAC99 model for administration of roles. *Proceedings of the 15th Annual Computer Security Applications Conference*. Phoenix, USA, IEEE Computer Society.

SANDHU, R. and MUNAWER, Q. (2002): A model for role administration using organization structure. *Proceedings of the seventh ACM symposium on Access control models and technologies*. Monterey, California, USA, ACM press.

SANDHU, R.S., COYNE, E.J., FEINSTEIN, H.L. and YOUMAN, C.E. (1996): Role-based access control models. *IEEE Computer* 29(2):38–47.

STEIN, L.A. (1987): Delegation is inheritance. *Proceedings of Object-Priented Programming Systems, Languages, and Applications (OOPSLA'87)*. New York, USA, ACM press.

TAMASSIA, R.,YAO, D.F and WINSBOROUGH, W.H.(2004): Role-based cascaded delegation. *Proceedings of the 9th ACM Symposium on Access Control Models and Technologies*. Yorktown Heights, New York, USA, ACM press.

YE, C.X., FU, Y.Q. and WU, Z.F. (2004): Research on user-role assignment based on role restrictive conditions. *Chinese Journal of Computer Science* 31(7):73–76.

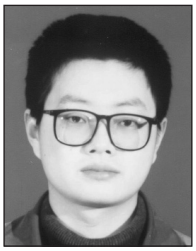
ZHANG, L.H., AHN, G.J. and CHU, B.T. (2001): A rule-based framework for role-based delegation. *Proceedings of 6th ACM Symposium on Access Control Models and Technologies (SACMAT)*. Chantilly, VA, USA, ACM press.

ZHANG, X.W., OH, S. and SANDHU, R. S. (2003): PBDM: A flexible delegation model in RBAC. *Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT'03)*. Como, Italy, ACM press.

ZHAO, Q.S., SUN, Y.F. and SUN, B. (2003): RPRDM: A repeated-and-part-role-based delegation model. *Chinese Journal of Computer Research and Development* 40(2):221–227.

## BIOGRAPHICAL NOTES

*Chunxiao Ye received his BEng in Computer Software from Sichuan University, China in 1995 and MEng in Computer Architecture from Chongqing University, China in 2002. He is a PhD student in the School of Computer Science at Chongqing University. His research interests include access control, database and software engineering*



Chunxiao Ye

*Zhongfu Wu received his BEng in Computer Science from Chongqing University, China in 1965. He is a professor of computer science in Chongqing University. His research interests include information security and network.*

Zhongfu Wu

*Yungqing Fu graduated from Chongqing University, China with a BEng in 1992, a MEng in 1995 and a PhD in 2002, all in Computer Science. He is an associate professor of computer science at Chongqing University. His research interests include CSCW, network and distance education.*

Yungqing Fu