An Attribute-Based-Delegation-Model and Its Extension

Chunxiao Ye¹, Zhongfu Wu¹, Yunqing Fu²

¹ College of Computer Science, Chongqing University, China
² College of Distance Education, Chongqing University, China

Abstract. 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 consists 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 his permissions to a person who does not satisfy the delegation constraint. To guarantee its flexibility and security, an extension of ABDM named ABDMX is proposed. In ABDMX, a delegator can delegate some high level permissions to low level delegatee candidates temporarily, but not permanently.

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) [1] security technology has gained considerable attentions [2] 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 [3]. Many studies have been done in delegation [4] [5] [6], and considerable attentions are paid to human-to-human delegation [3] [7] [8].

But there are still some problems in delegation need to be solved: 1. Because delegation is controlled by delegator itself, a malicious user can delegate some important permissions to low level delegatees. 2. The Delegation security relies heavily on system 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 be 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 ABMD, \(ABDM_X\) and some existing delegation models. Conclusions and future works are presented in section 6.

## 2 Related Works

RBDM [7] [8] 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 [9] to specify and enforce policies in delegation. The delegation unit in RBDM and RDM2000 is “role”. In RPRDM [10], a delegator can delegate part of his/her permissions to a delegatee by a “mask”.

PBDM [3] 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 permissions to delegatees. In PBDM1 and PBDM2, the permission flow is managed by a security administrator with delegatable role (DBR). RDM2000 and PBDM can be seen as special cases of PBDM.

In most cases, a delegator cannot delegate all of his/her 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 himself. RPRDM only addresses repeated and partial delegation, and delegation in RPRDM is also controlled by the 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 delegates, but the prerequisite condition in these models consists only of prerequisite role or organization unit [11] [12] [13]. RBAC and other delegation models overlook the differences between 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 [14] [15] [16]. In RB-RBAC [15] [16], users who have attribute expression will be assigned to roles dynamically and automatically. Attribute expression in [17] indicates the user’s qualification or ability required by a role.
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 or abilities. This is a decided-delegatee delegation. In other case, the FM may want to delegate some permissions to a user who has the required qualifications or abilities, but he does not know who has the required qualifications or 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 secured 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 his/her permissions to tdr. Finally, he/she 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 temporary delegation role.

The delegation prerequisite condition in our delegation model consists of both prerequisite condition (CR) [9] and delegation attributes 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** An attribute expression, uae, is of the form ua roprt uav, where ua is an attribute specified by system, roprt is an operator in {<, ≤, >, ≥, =, ≠} and uav is an attribute value specified by system.

For examples, level=4, type=’S’, and total≥33 are uae.

**Definition 2** uae and uae are said to have identical structures if and only if they have the same uas and roprts. uae and uae are said 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 [15] [16], we use the symbol ‘≥’ to denote the dominance relations between two uae. Here we also propose a method which is an extension of those in recent studies [15] [16] to judge the dominance relation between two comparable uae:

- Suppose two uae have the form of ua ≥uav or ua > uav:
  - If uav is a numeric value, then the relation of ‘≥’ automatically follows the normal order of uavs.
  - If uav is not a numeric value, then the relation of ‘≥’ must be manually specified.
suppose two uaes have the form of \( ua \leq uav \) or \( ua < uav \)
- If \( uav \) is a numeric value, then the relation of ‘\( \geq \)’ goes in reverse order of \( uav \).
- If \( uav \) is not a numeric value, then the relation of ‘\( \geq \)’ must be manually specified.

Suppose two uaes have the form of \( ua = uav \) or \( ua \neq uav \), the relation of ‘\( \geq \)’ must be manually specified.

For example, we can say \( uae_1 (level>5) \geq uae_2 (level>4) \) and \( uae_3 (total \leq 20) \geq uae_4 (total \leq 20) \). 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.

Definition 3 A DAE is a delegation attribute expression using AND on terms of the form \( uae \) where AND is the logic operator ‘and’ and \( uae \) is an attribute expression.

For examples, \( level=4, type='S', \) and \( total \leq 20 \) AND \( type='S' \) are DAEs.

In some of the existing models [15] [16], 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. User’s DAE indicates a user’s status, ability and qualification, while permission’s DAE indicates a delegatee’s ability or qualification required by this 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.

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’ DAE 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 a uae set of a DAE. For example, the \( UAE \) of \( level>5 \) AND \( total \leq 20 \) is \{ \( level>5 \), \( total \leq 20 \) \}.

We use ‘\( \triangleright \)’ to denote the dominance relation between two DAEs:

Definition 4 We say \( DAE_1 \triangleright DAE_2 \), 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 set 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 \( total \leq 20 \) \( \triangleright \) \( DAE_2 \) (\( level>4 \) AND \( total \leq 30 \)) for \( level>5 \geq level>4 \) and \( total \leq 20 \gtrless total \leq 30 \). We can also say \( DAE_1 \) (\( level>5 \) AND \( total \leq 20 \) \( \triangleright \) \( DAE_4 \) (\( level>4 \)) according to definition 4.

We can say a user is a qualified delegatee of \( tdr \) if his/her DAE \( \triangleright tdr.DAE \) in delegation. Here we introduce a DAE generation algorithm named DG algorithm as below:

DAG (DAE Generation) Algorithm:

Input: \( p \in P \), where \( P \) is the permission set of \( tdr \).
Output: DAE of \( tdr \)

Begin

\( UAE = \emptyset \);

for \( i = 1 \) to \( n \)
UAE = UAE ∪ UAE_i, where UAE_i is the uae set of p_i.DAE

for i = 1 to |UAE|
    for j = 1 to |UAE|
        if uae_i ≠ uae_j and uae_i ≥ uae_j then delete uae_j from UAE

Return DAE = uae_1 AND...AND uae_n, where uae_1,..., uae_n ∈ UAE, n = |DAE|

End

In 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 DG algorithm can reflect the comprehensive requirements of users’ DAEs required by delegation permissions and thus guarantee the security of delegation.

3.2 ABDM

Definition 5 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 = RR × RR is a regular role hierarchy

• TDRH_u = TDR × TDR is a temporary delegation role hierarchy owned by a user u

• R = RR ∪ TDR

• RR ∩ TDR = Φ

• URA = U × RR is a user to regular role assignment relation

• UDA = Ude × TDR is a decided-delegatee to temporary delegation role assignment relation

• UEA = Uee × TDR is a undecided-delegatee to temporary delegation role assignment relation

• UA = URA ∪ UDA ∪ UEA

• PRA = P × RR is a permission to regular role assignment relation

• PDA = P × TDR is a permission to temporary delegation role assignment relation

• roles: U → 2^R is a function mapping a user to a set of roles

• per_r: RR → 2^P is a function mapping a regular role to a set of permissions

• per_d: U ∪ TDR → 2^P is a function mapping a temporary delegation role to a set of permissions
per_d(u,tdr) = \{p| (\exists tdr' \leq tdr)((p, tdr') \in PDA) \wedge tdr' \in roles(u)\}

per_u: U \rightarrow 2^P is a function mapping a user to a set of permissions

\text{per}_u(u) = \{p| (\forall r \in RR)((u, r) \in URA) \cup \{p((\exists r \in TDR)((u, r) \in UDA) \wedge (p, r) \in PDA)\} \cup \{p| (\exists r \in TDR)((u, r) \in UDA) \wedge (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| (\forall p \in \text{per}_d(u, tdr))(p \notin \text{per}_u(u)) \wedge (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.DAE tdr.DAE \wedge (\forall p \in \text{per}_d(tdr))(p \notin \text{per}_u(u))\}

can-delegateD \subseteq R \times CR \times DAE \times TDR is a delegation constraint on UDA

can-delegateU \subseteq R \times CR \times Uee \times TDR is a delegation constraint on UEA.

For example, \text{can-delegateD}\{ST, TR, level=4 \text{ and type}='S' \text{ and total}=35, tdr\} means that a delegator who has ST can assign a delegatee who must has role TR and his DAE satisfies level=4 \text{ and type}='S' \text{ and total}=35 to tdr. \text{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 qualified delegatees set of tdr and alex has role TR.

Fig. 1. ABDM model

Here some examples are given to show how ABDM works. Let us discuss the case in figure 2. For convenience of understanding, we suppose delegatees do not have the same permissions as those of tdr before delegation. Figure 2 also gives an example of role hierarchy, user’s DAE and its roles, and permission’s DAE. Tom with a role ST is supposed to want to delegate his permissions \{Borrow in S, Read in S, 5 books one time\} to someone. First, he must create a temporary delegation role tdr. Second, he can assign permissions \{Borrow in S, Read in S, 5 books one time\} to
The tdr’s DAE now is generated by DG algorithm with the input of DAEs of Borrow_in_S, Read_in_S and 5 books one time.

In ABDM, the system can automatically generate a Uee(tdr) with qualified delegate 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. Delegation failed for neither Annie nor Lucy is a qualified delegate.

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. This time, Uee(tdr)= {Alex, John, Mike}.
2. Tom assigns Alex to tdr. Delegation is successful if Alex has role TR, otherwise delegation 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.

Fig. 2. Example of ABDM
In the case in Table 1, for example, if a teacher \( t \) (has the role teacher) wants to delegate \( p_1 \) and \( p_2 \) to a student \( s \), he/she must first create a temporary delegation role \( tdr \), and then assigns \( p_1, p_2 \) to it. Now, \( tdr.DAE \) is type \( T \). In this case, suppose \( DAE: type='T' \) \( \triangleright \) \( DAE: type='S' \) and all students have the same \( DAE: type='S' \). \( t \) cannot perform a decided-delegatee delegation, for \( s \) is not a qualified delegate of \( tdr \). Then \( t \) tries to perform an undecided-delegatee delegation. Because none of the students satisfy \( tdr \), he/she cannot delegate \( p_1, p_2 \) to a student in an undecided-delegatee delegation either.

**Table 1: Permissions and permissions’ DAE**

<table>
<thead>
<tr>
<th>permissions of teacher</th>
<th>permission’s DAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_1 ): reading in the teacher’s reading room</td>
<td>Type='T'</td>
</tr>
<tr>
<td>( p_2 ): borrow books from teacher’s reading room</td>
<td>Type='T'</td>
</tr>
<tr>
<td>( p_3 ): create exams</td>
<td>Type='T'</td>
</tr>
<tr>
<td>( p_4 ): record results</td>
<td>Type='T'</td>
</tr>
</tbody>
</table>

In fact, there are some differences between \( p_1, p_2 \) and \( p_3, p_4 \): \( p_1, p_2 \) can be temporarily delegated to a person who has not the required abilities or qualifications. It will not cause any security problems. But they cannot be permanently delegated to an unqualified person, for that will go against security policy. \( p_3 \) and \( p_4 \) can be delegated to a person if he/she has the required abilities or qualifications both in a temporary and permanent delegation. So, a person with a role teacher can delegate his/her permissions \( p_1 \) and \( p_2 \) to a person temporarily but he/she cannot temporarily delegate his/her permissions \( p_3 \) and \( p_4 \) to a person in any cases. That is, \( p_1 \) and \( p_2 \) can be delegated to a low level person temporarily but not permanently.

### 4.1 ABDMX

To overcome this shortcoming, we introduce a model named ABDMX, 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. So the delegator can temporarily delegate NMP to a low level delegatee candidate.

**Definition 6** 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:

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

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 in advance.
**Definition 7** A user \( u \)'s temporary delegation role \( tdr \) is a monotonous role if it has an identical restriction on delegatee’s \( DAE \) both in a temporary and a permanent delegation. \( tdr \) is a non-monotonous role if it has restriction on delegatee’s \( DAE \) only in a permanent delegation. \( MN \) (\( tdr \)) is a function defined as follows:

\[
MN \left( u, tdr \right) = \begin{cases} 
\text{True} & \exists \ p \in \text{per} \_ \_d \left( u, tdr \right), \\
MN \left( p \right) = \text{True} \\
\text{False} & \forall \ p \in \text{per} \_ \_d \left( u, tdr \right), \\
MN \left( p \right) = \text{False}
\end{cases}
\]

That is, a non-monotonous role has no restriction on delegatee’s \( DAE \) in a temporary delegation.

Because ABDM does not support delegation with NMPs, we must modify it to meet this requirement.

**Definition 8** The following is a list of ABDM3 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, MPs, NMPs, 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
- \( UEA \subseteq Uee \times TDR \) is an undecided-delegatee to temporary delegation role assignment
- \( UDA = UDAM \cup UDAN \)
- \( UA = URA \cup UDA \cup UEA \)
- \( PRA \subseteq P \times RR \) 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
  \[
  \text{roles} \left( u \right) = \{ r \mid (u, r) \in UA \}
  \]
- **per_r**: \( RR \rightarrow 2^P \) is a function mapping a regular role to a set of permissions
\[ \text{per}_r(r) = \{ p | \exists r' \leq r \in \text{PRA} \} \]

- \text{per}_d: U \cup \text{TDR} \to 2^p \] is a function mapping a temporary delegation role to a set of permissions
  \[ \text{per}_d(u, tdr) = \{ p | \exists r \leq tdr \in \text{PRA} \} \]

- \text{per}_u: U \to 2^p \] is a function mapping a user to a set of permissions
  \[ \text{per}_u(u) = \{ p | r \in \text{RR} \} \]

- \text{Ude}: \text{TDR} \to 2^u \] is a function mapping a temporary delegation role to a set of users
  \[ \text{Ude}(tdr) = \{ u | p \notin \text{per}_d(u, tdr) \} \]

- \text{Uee}: \text{TDR} \to 2^u \] is a function mapping a temporary delegation role to a set of qualified users
  \[ \text{Uee}(tdr) = \{ u | p \notin \text{per}_d(u, tdr) \} \]

\[ \text{can-delegateM} \subseteq R \times CR \times \text{DAE} \times \text{TDR}_M \] is a constraint on \text{UDAM}

\[ \text{can-delegateN} \subseteq R \times CR \times \text{TDR}_N \] is a constraint on \text{UDAN}

\[ \text{can-delegateU} \subseteq R \times CR \times \text{Uee} \times \text{TDR}_M \] is a delegation constraint on \text{UEA}

**Fig. 3. ABDM model**

\[ \text{can-delegateN(r, cr, tdr)} \] means a delegator with role \( r \) can delegate a non-monotonic temporary delegation role \( tdr \) to a delegatee who has role \( cr \).

\[ \text{can-delegateM} \] can restrict delegates with temporary delegation role’s \( \text{DAE} \) and delegation permission’s monotony need not to be considered, on the contrary it must be considered in \( \text{can-delegateN} \) for it cannot restrict delegatee with temporary
delagation roles’ DAE. That is, can-delegateN only be used in a delegation with NMPs while can-delegateM can be used in a delegation with both MPs and NMPs. Because a NMP has no restriction on a delegatee’s DAE in delegation, ABDMX does not support undecided-delegatee delegation with NMPs. We can prevent ABDMX from generating Uee (tdr) for NMPs by adding a constraint: MN (tdr) = True to the definition of it.

Let us discuss the example presented in section 4 again to show how this extended model works. In one case, teacher t wants to delegate his permissions p1, p2 to a student s. He/she can delegate them according to the following steps (in table 1, p3, p4 are MPs and p1, p2 are NMPs):
1. t creates a temporary delegation role tdr.
2. t assigns p1, p2 to tdr. That is, MN (tdr) = False for MN (p1) = False and MN (p2) = False.
3. t must perform delegation by UDAN for tdr ∈ TDRN.
4. Delegation is successful for UDAN (s, tdr) satisfies can-delegateN (teacher, student, tdr) constraint.

In the other case, t wants to delegate his/her permissions p2, p3 to s:
1. t creates a temporary delegation role tdr.
2. t assigns p2, p3 to tdr.
   That is, MN (tdr) = True for MN (p1) = False and MN (p2) = True and tdr’s DAE is type = "T".
3. Delegator must perform delegation by UDAM for tdr ∈ TDRM.
4. Delegation failed because UDAN (s, tdr) does not satisfy can-delegateN (teacher, student, tdr) constraint.

Undecided-delegatee delegation with MPs and delegation evocation in ABDMX are similar to those in ABDM. Revocation in ABDMX is similar to that in ABDM.

4.2 Delegation security in ABDMX

We now discuss delegation security in ABDMX according to a temporary delegation role tdr’s monotony

1. MN (tdr) = True
   In this case, tdr has MPs and the delegator can perform delegation by UDAM. Because can-delegateM is the delegation constraint on UDAM, a delegatee must be a qualified one when assigned to tdr.
   A delegator can delegate some NMPs to delegatees by UDAM. That means a delegatee has those prerequisite roles and DAE which are required by can-delegateM when assigned to tdr. This will not cause any security problems.
2. MN (tdr) = False
   In this case, tdr has NMPs and the delegator can perform delegation with UDAN. Although can-delegateN cannot restrict delegatees with DAE, there will be no security problem in the delegation. The reason is that in fact NMPs have no restrictions on delegatees in this type of delegation.
A delegator cannot delegate MPs to delegatees by $UDAN$. In ABDM$_X$, $can-delegateN$ constraint means only NMPs can be delegated to delegatees by $UDAN$ ($tdr \in TDR_N$). This will not cause any security problems in delegation either.

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 role. 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 he/she is 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 NMPs to an unqualified low level user but cannot temporarily delegate MPs to an unqualified delegatee in any case. In fact ABDM$_X$ does not cause any security problems in temporary delegation for NMPs have no restrictions on delegatee candidates’ $DAE$s. So, ABDM$_X$ has a medium flexibility but the same security level as that of ABDM in delegation.

![Fig. 4. Security and Flexibility of ABDM, ABDM$_X$, PBDM, RDM2000 and RBDM.](image)

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. ABDMX is more flexible than ABDM in delegation. For in ABDMX, a delegator can temporarily delegate NMPs to low level users without causing any security problems. Both ABDM and ABDMX can be used in temporary and permanent delegation and make delegation securer and more flexible.

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

Acknowledgments

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

References