An Auto-revocation Supported Delegation Model

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

Xiang Li
Chongqing survey institute, China
Email : ycxalex@gmail.com

***Abstract*—We have proposed an Attribute-Based Delegation Model (ABDM), in which delegatee must satisfy both delegation prerequisite condition (CR) and delegation attribute expression (DAE) when assigned to a delegation role. ABDM introduce some auto revocation mechanics to support two new types of auto revocation: revocation triggered by the change of user's delegation attribute expression and revocation triggered by the change of delegated permission's delegation attribute expression, which are different from existing revocations. ABDM supports auto revocation triggered by time. This paper also discusses system cost of auto revocation and security of multi-step delegation. An auto revocation algorithm and a system architecture are proposed in the end of this paper.**

***Index Terms*— information security, access control, delegation, auto revocation, attribute**

I. INTRODUCTION

Role-Based Access Control (RBAC) [1] has emerged as a new security technology which is an alternative to DAC and MAC. RBAC has recently received considerable attention. In RBAC, permissions are associated with roles, and users are assigned to appropriate roles to acquire permissions.

Delegation means that a user (delegator) can give his permissions to someone (delegatee). There are three types of situations in which delegation can take place: backup of roles, decentralization of authority and collaboration of work. Human-to human delegation has received considerable attention recently [2, 3, 4, and 5].

Revocation means a system administrator can revoke membership of a user from regular role. There exist weak and strong revocations according to explicit and implicit memberships. These revocations are mainly used in the management of user-role assignment [6].

RBDM [7] is the first delegation model based on role. RBDM addresses human-to-human delegation. It deals with flat and hierarchical role, multi-step delegation. In RBDM, a user who owns a role can delegate his role to another user. Revocation of RBDM can be manually performed by delegator and system administrator. Revocation also can be automatic triggered by delegation duration.

RDM2000 [6] is an extension of RBDM. A rule-based declarative language has been proposed to specify and enforce policies in delegation. The unit of delegation in RBDM and RDM2000 is "role". Only user revocation is considered in RDM2000 and there are two types of user revocation: grant-dependent and grant-independent revocation. User revocation in RDM2000 has two options: non-cascading and cascading revocation. RDM2000 proposes three rules as enforcement of delegation revocation policies.

PBDM [8] is a flexible delegation model that supports multi-step delegation and revocation in role and permission level. PBDM includes three sub-models:PBDM0, PBDM1, and PBDM2. 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 delegatable role (DBR). Revocation in PBDM is seen as the reverse process of delegation and a delegator can remove his/her own delegation at any time. PBDM supports multi-step revocation but revocation is discussed shortly.

RPRDM [9] only addresses repeated and partial delegation. There are four types of revocations in RPRDM. An original delegator, system administrator or a delegated delegator can revoke delegated permissions from delegatees. But there is only a very short discussion on automatic revocation.

A family of models called RB-RBAC allowing the specification of automatic (implicit) user-role assignment has been introduced in [10]. Revocation of can-assume relation in RB-RBAC is similar to the revocation in ABDM but can-assume isn't a delegation relation. Revocation of can-delegate in RB-RBAC can be achieved in four ways but none of which relates to rules or attribute expressions.

QBCDM [11] discussed revocation and supports eight types of revocation, but none of them can revoke delegation permissions or roles automatically. In TRDM [12], two revocations were discussed: revocation performed by user and system automatically. But it has not elaborate how auto revocation really works. The author discussed how to perform a multi-granularity user to user cascade delegation in [13]. It also discussed cascade revocation, but has little on auto revocation.

We have proposed a new delegation model named Attribute-Based-Delegation-Model (ABDM) in [14]. Delegation constraint in our delegation model includes delegation prerequisite condition (CR) and delegation attribute expression (DAE). ABDM is a strict and more secured delegation model both in temporary and permanent delegation. We have not elaborated revocation in [14] for it is only focus on attribute based delegation.

doi:10.4304/jcp.5.6.861-869

In this paper, we introduce four new types of automatic revocation: revocation triggered by delegation duration, the changes of delegatees' or delegated permissions' DAEs and revocation triggered by CR. In these types of revocation, system can remove delegated permissions from delegatees automatically according to system time, changed DAEs and CRs. We also discuss some revocation modes of automatic revocation.

The rest of this paper is organized as follows. Section II first introduces some important concepts, such as attribute and attributes expression, and then presents basic ABDM model and its extension on user and administrator revocation. Section III introduces auto revocation in ABDM model. Section IV presents system implementation and case study. Conclusions and future works are presented in section V.

II. ABDM MODEL

A.Concepts

We introduce user, permission attribute, and attribute expression in this section.

Definition 1 DAE::=uae|
uae AND uae|
(uae AND uae)
uae::=ua roprt uav
roprt::= <|≤|>|≥|=|≠
ua::={specified by system}
uav::={specified by system}

ua, uav, uae, DAE are attribute, attribute value, attribute expression, delegation attribute expression, respectively. AND is the logical "and" operation.

For examples, level=4, level=5, type='S', type='J', total≥20 and total≥33 are uaes. level=4 AND type='J' AND total≥20 is a DAE, and level=5 AND type='S' AND total≥33 is a DAE too.

Definition 2 AVD::=[startime, endtime], where startime and endtime are system time. The minimum value of startime is 1900/1/1 0:0:0 and the maximum value of endtime is 9999/12/31 23:59:59.

As mentioned in recent studies [14], we also use '▷' to denote dominance:

Definition 3 uae_i uae_j are said comparable only if they have identical structures. Only comparable uaes are tested for dominance.

If uae_i and uae_j have the form: ua ≥uav or ua > uav:

1. uav is numeric value, '▷' automatically follows the normal order of values.

2. uav is not numeric value: '▷' must be manually specified.

If uae_i and uae_j have the form: ua ≤ uav or ua < uav

3. uav is numeric value, '▷' goes in reverse order of values.

4. uav is not numeric value: '▷' must be manually specified.

In case of inequality or equality, '▷' must be manually specified.

We can say level=5 ▷ level=4, total≥33 ▷ total≥20 according to case 1. type='S' ▷ type='J' is manually specified according to case 2.

Definition 4 two DAEs are said to be comparable only if they have identical structures. Only comparable DAEs are tested for dominance. We say DAE_i ▷ DAE_j, if $\forall$ $uae_i \in DAE_i$, $uae_j \in DAE_j$, uae_i ▷ uae_j.

For example, DAE_1 (level=5 AND type='S' AND total≥33) ▷DAE_2 (level=4 AND type='J' AND total≥20) according to definition 2 and 3.

In some existing models [7, 10], only user has attribute expression and attribute expression is a part of rule in URA. The main improvement of our work is: both users and permissions in ABDM have DAEs. User's DAE indicates user's current states, abilities and qualifications. Permission's DAE indicates delegatee's abilities, qualifications and current states that are required by permission in delegation. Permission's DAE is a part of delegation constraint and can be used to generate a temporary delegation role's DAE.

In our model, delegator must first create a delegate role named temporary delegation role, say tdr. Then, delegator can assign his/her permissions to tdr. Finally, he/she can assign users to tdr.

A temporary delegation role (tdr) has its own DAE, which is a part of delegation constraint. Because all permissions in tdr have DAEs, tdr.DAE is a combination of those permissions' DAEs. If a permission's DAE dominates another permission's DAE, the latter's DAE will be replaced by the DAE of the former in tdr.DAE. For example, supposing tdr has only two permissions: p1 and p2, where p1.DAE is 'age≥20' and p2.DAE is 'age≥30'. The final tdr.DAE is 'age≥30' for p2.DAE dominates p1.DAE and p1.DAE has been replaced by p2.DAE. tdr.DAE is automatic generated by system before delegation.

Delegation constraint in our delegation model includes both prerequisite condition (CR) and delegation attributes expression (DAE). Only the one who has those prerequisite roles and his DAE dominates a temporary delegation role's DAE can be assigned to that temporary delegation role. Delegation attribute expression and prerequisite condition are a more strict constraint in delegation.

In ABDM, there are two types of delegations: decided-delegatee and undecided-delegatee. In a decided-delegatee delegation, delegator must specify delegatee candidates manually before delegation. In an undecided-delegatee delegation, delegatee candidates are generated automatically by system.

B.The Model

Definition 5 the following is a list of main ABDM components:

- AR, R, RR, TDR, S, P, U, Ude, and Uee are set of administrative roles, 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.
- $TDRH_u \subseteq TDR \times TDR$ 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.
- UARA ⊆ U×AR is a user to administrative role assignment relation.
- UDA ⊆ Ude×TDR is a decided-delegatee to temporary delegation role assignment relation.
- UEA ⊆ Uee×TDR is an 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: $U \rightarrow 2^R$ is a function mapping a user to a set of roles.
  roles (u) = {r| (u, r) ∈ UA}
- per_r: $RR \rightarrow 2^P$ is a function mapping a regular role to a set of permissions.
  per_r(r) = {p|(∃ r'≤r) (p, r') ∈ PRA}
- per_d: $TDR \rightarrow 2^P$ is a function mapping a temporary delegation role to a set of permissions.
  per_d(tdr)={p|(∃ tdr'≤tdr)((p, tdr') ∈ PDA)}
- per_u:$U \rightarrow 2^P$ is a function mapping a user to a set of permissions.
  per_u(u)={p|(∃ r ∈ RR)((u,r) ∈ URA∧(p, r) ∈ PRA)}∪{p|(∃ r ∈ TDR)((u,r) ∈ UDA∧(p, r) ∈ PDA)}∪{p|(∃ r ∈ TDR)((u,r) ∈ UEA∧(p, r) ∈ 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|(∀ p∈ per_d(tdr)) (p∉ per_u(u))∧(u,tdr) ∈ 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∧(∀ p∈ per_d(tdr)) (p∉ per_u(u))}
- can-delegateD ⊆ R×CR×DAE×TDR×AVD is a delegation constraint on UDA.
- can-delegateU ⊆ R×CR×Uee×TDR ×AVD is a delegation constraint on UEA.
- CR: $R \rightarrow 2^r$ is a function mapping a role to a set of roles.
  CR (r) = {r'|r' ∈ RR∧r' is a prerequisite role of r in a URA}

Fig. 1 gives the main components and of ABDM. Here only lists URA and PRA relations, and revocation relations are not shown in fig. 1.

To make ABDM a complete model, we extend it with two types of manual revocation:

- can-revokeU⊆U×RR×TDR×U
  (u,r,tdr,u') ∈ can-revoke⇔ delegated(u,tdr,u') =true ∧ (u, r) ∈ URA

can-revokeU is a relation of user revocation. That means: a user u who owns role r can revoke tdr from user u', to whom u has delegated tdr.

- can-revokeA⊆U×AR×TDR×U
  (u,r,tdr,u') ∈ can-revoke⇔ (u, r) ∈ UARA

can-revokdeA is a relation of administrator revocation. That means: an administrator u who has a administrive role r can revoke tdr from u'.

So, with user and administrator revocation, delegator or system administrator can remove delegated permissions from delegatees when a delegation ends. These two types of revocation must be performed manually. It is easy to implement and make delegation simply.

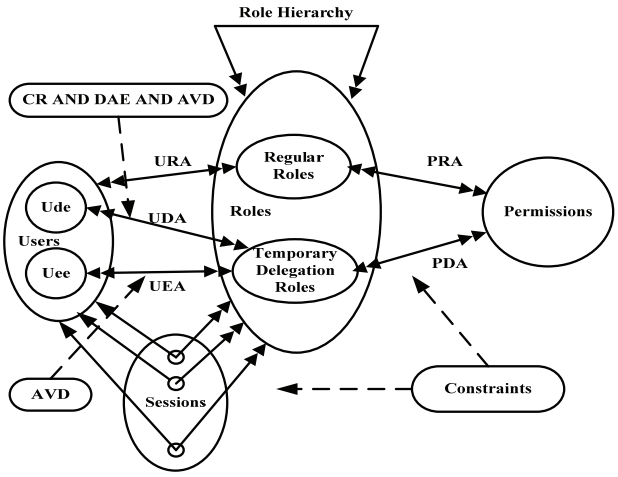

Figure 1. ABDM model

III. AUTO REVOCATION

Although user and administrator revocation can revoke all delegated roles from delegatee, they have some shortages:

- labor intensive

Because all delegation must be operationed manually, it will increase the delegation management burden on both users and administrators, especially delegator who has performed many delegations. The administrator will also suffer heavy burden in a large system with thousands and thousands of users.

- low security

Based on delegation attribute expression, ABDM is a model can improve the delegation security. On one hand, because DAE is a vital constraint in delegation, any change of user or permission attribute expression may cause the violation of some delegation constraints. So, if those delegations must be revoked manually and have not revoked immediately, there will exist many delegations in which delegatees may not satisfied those required abilities or qualifications.

On the other hand, CR contains roles that are needed for a user to active a delegation role. So, a delegation role must be revoked immediately from a delegatee if the CR of the delegatee has changed and the CR has not enough roles to support delegation role.

The original ABDM is not safe for it does not support auto-revocation in these two situations.

Here we introduce for types of auto revocation as follows:

A. Revocation Triggered by Delegation Duration

Definition 6 we say delduration(u,tdr) is valid if curtime() is prior to deleduration(u,tdr), denoted by $\uparrow$ delduration(u,tdr); otherwise, delduration(u,tdr) is invalid and denoted by $\downarrow$ delduration(u,tdr)

Definition 7 can-auto-revoke-caused-by-time$\subseteq$U×TDR, (u, tdr) $\in$can-revoke-caused-by-time $\Leftrightarrow$(tdr$\in$roles (u)) $\wedge \downarrow$ delduration (u,tdr))

In this type of revocation, when the delegation duration of a delegation in which a user u was delegated a delegation role tdr is invalid, tdr will revoke from u automatically.

ABDM uses DAE as a vital constrain in delegation and it require that user's DAE must satisfies temporary delegation role's DAE to secure a success delegation. Obviously, if a user's DAE will not satisfies a tdr's DAE anymore, that means the user has not the abilities or qualifications to active the tdr, so the tdr must be revoked from the user immediatelly. Because this type of revocation is based on attribute expression and it's change, the change of attribute expression can be used as an important triggering mechanism for auto-revocation.

There are three situations that a user's DAE (say, u.DAE) does not satisfiy a temporary delegatin role's DAE (say, tdr.DAE):

1. u.DAEs changed and tdr.DAEs remain. In this situation, maybe the changed u.DAEs do not satisfy unchanged tdr.DAEs.
2. u.DAEs remains and tdr.DAEs changed. In this situation, maybe the unchanged u.DAEs do not satisfy the changed tdr.DAEs.
3. u.DAEs and tdr.DAEs all changed. In this situtation, maybe the changed u.DAEs do not satisfy the changed tdr.DAEs.

Here we discuss two types of auto-revocation trigged by changed attribue expressions:

B. Revocation Triggered by User's Delegation Attribute Expression

There exist two possibilities in this type of revocation: the changed u.DAE still satisfies the delegated permission's DAE or not. In the first possibility, revocation can not be triggered but in the last one, revocation must be triggered.

The change of tdr.DAE can be caused by the change of p.DAE, for the delegated permissions are assigned to temporary delegation role (denoted as tdr), so. A revocation can be triggered by judging whether an u.DAE satisfies the changed tdr.DAE (denoted as u.DAE $\triangleright$ tdr.DAE) or not (denoted as $\neg$u.DAE $\triangleright$ tdr.DAE).

To describe revocation triggered by the change of user's delegation attribute expression, here we first define a function to judge whether a user's delegation attribute expression has been changed of not:

Definition 8

$$\text{DAEModified}(u)=\begin{cases}\text{true} & \text{u.DAE has been changed}\\ \text{false} & \text{u.DAE has not been changed}\end{cases}$$

In this type of revocation, if a changed u.DAE can not satisfy a tdr.DAE, a revocation can be triggered automatically by system. This operation can be restricted by following relation:

Definition 9 revocation triggered by the changed u.DAE must be restricted by: can-revoke-caused-by-delegatee $\subseteq$ U×TDR,

(u,tdr)$\in$can-revoke-caused-by-delegatee$\Leftrightarrow$(tdr$\in$roles(u))$\wedge\neg$u.DAE$\triangleright$tdr.DAE)$\wedge$(DAEModified(u)=true), where U is the set of user and TDR the set of temporary delegation role. Functions in this definition can be found in section II.

Definition 8 means, if a changed u.DAE can not satisfy a tdr.DAE and u has been delegated tdr, tdr must be revoke from u automatically.

C. Revocation Triggered by Permission's Delegation Attribute Expression

In this case, if a delegatee's DAE can no longer dominate a changed DAE of a delegated permission, he or she can't keep this delegated permission any more. On the contrary, a delegatee can continue keep a delegated permission with a changed DAE if the delegatee's DAE still dominates this permission's new DAE.

In ABDM, a temporary delegation role's DAE must be automatic regenerated immediately if the DAE of a delegated permission in a temporary delegation role has been changed. So, there may have some differences between the changed DAE of a temporary delegation role and its original DAE. These differences may trigger the automatic revocation.

For a better understanding, supposing tdr is a temporary delegation role and u is a delegatee who has been assigned to tdr. Pi is a delegated permission in tdr and its DAE has been changed.

1. u.DAE still dominates tdr.DAE

In one case, although pi's DAE has been changed, it has been replaced by a dominant delegated permission's DAE in the regenerated tdr.DAE. So, tdr.DAE hasn't been changed and u.DAE still dominates tdr.DAE. In another case, tdr.DAE has been changed due to the change of pi.DAE but u.DAE still dominates tdr.DAE. In both cases, u can still have tdr.

2. u.DAE doesn't dominate tdr.DAE

In this case, tdr.DAE has been changed due to the change of pi.DAE and u.DAE doesn't dominate tdr.DAE any more. So, u will be revoked from tdr automatically.

There also exist two situations when a permission's DAE has been changed: a user's DAE still satisfies the changed p.DAE or a user's DAE can not satisfy the changed p.DAE any more. Here we also define a function to judge whether a p's DAE has been changed or not.

Definition 10

$$\text{DAEModified}(p)=\begin{cases}\text{true} & \text{p.DAE has been changed}\\ \text{false} & \text{p.DAE has not been changed}\end{cases}$$

So, in this type of revocation, system can only judge wither the u.DAE satisfies the tdr.DAE or not to automatically trigger the revocation or not.

Similarly, we can define following relation to restrict this type of revocation:

Definition 11 revocation triggered by permission's delegation attribute expression must be restricted by:

can-revoke-caused-by-permission $\subseteq$ U×P×TDR,

(u,p,tdr)$\in$can--revoke-caused-by-permission $\Leftrightarrow$(tdr$\in$roles(u))$\wedge\neg$(u.DAE$\triangleright$tdr.DAE)$\wedge$(p$\in$per_d(tdr)) $\wedge$(DAEModified(p)=true), where U is the set of user, P the set of permission and TDR the set of temporary delegation role. This definition means, if the DAE of a tdr's permission has been changed and the user u that has tdr and u's DAE can not satisfy tdr.DAE any more, tdr can be automatically revoked from u.

D. Revocation Triggered by CR

In RBAC, system administrator use CR as a constraint in URA. That means, CR is a set of roles which are needed by another role. Because user to user delegation can be seen as a kind of URA, CR is also a constraint in delegation. So, if the CR of a delegation has changed, maybe the delegation will not satisfy this constraint. The delegation must be revoked immediately.

Similar to above woks, we first define a function as follows:

Definition 12

$$\mathrm{CRModified}(u)=\begin{cases}\text{true} & \text{u.CR has been changed}\\ \text{false} & \text{u.CR has not been changed}\end{cases}$$

In this type of revocation, if a changed u.CR does not satisfy a CR (u,tdr), a tdr can be revoked automatically by system. This operation can be restricted by following relation:

Definition 13 revocation triggered by the changed u.CR must be restricted by: can-revoke-caused-by-CR $\subseteq$ U×TDR,

(u,tdr)$\in$can-revoke-caused-by-CR$\Leftrightarrow$ (tdr $\in$ roles(u)) $\wedge\neg$u.CR$\triangleright$ CR(u, tdr))$\wedge$(CRModified(u,tdr)=true), where U is the set of user and TDR the set of temporary delegation role. Functions in this definition can be found in section II.

E. Revocation Modes

There exist some revocation modes in automatic revocation:

- instant/dilatory automatic revocation

Instant automatic revocation means when a delegatee's DAE doesn't dominate a temporary delegation role's DAE, the delegatee must be revoked from this temporary delegation role immediately. Instant automatic revocation can have two options: non-cascading instant automatic revocation and cascading instant automatic revocation. Cascading instant automatic revocation means revokes a node from a delegation tree together with the sub-tree blow the node, while non-cascading instant automatic revocation only revokes the node. Cascading instant automatic revocation sometimes is difficult to implement. Here we only consider non-cascading instant automatic revocation and let cascading instant automatic revocation be our future work.

In dilatory automatic revocation, if a delegatee's DAE doesn't dominate a temporary delegation role's DAE, the delegatee can still have this temporary delegation role until the temporary delegation role has been deactivated. For example, if a delegatee hasn't finished his or her task, which requires those delegated permissions and these delegated permissions' DAEs have been changed, an automatic revocation thus can be triggered subsequently. But these delegated permissions must not be revoked from delegatee until he or she has finished the task.

Instant automatic revocation is a default revocation mode in automatic revocation, for dilatory mode has two shortcomings:

1. Supposing a delegatee u, whose current membership includes a temporary delegation role tdr. In the dilatory automatic revocation, u can further delegate tdr's permissions to other users. This is illegal in ABDM for this time u isn't a qualified delegatee any more and he has no right to perform a cascading delegation with tdr. Also, it can make delegation and revocation management more complicated.
2. Dilatory automatic revocation must be triggered by the change of temporary delegation role's activity state. This means ABDM must record every temporary delegation role's current activity state. It will increase the complexity of temporary delegation roles management.

- total/partial automatic revocation

Total automatic revocation means if a delegatee's DAE doesn't dominate a temporary delegation role's DAE, all of this temporary delegation role's permissions must be removed from the delegatee automatically. Total automatic revocation is also a default revocation mode in ABDM and it can eliminate the overhead of administrative effort in revocation.

In the partial automatic revocation, if a delegatee's DAE doesn't dominate a temporary delegation role's DAE, only those permissions whose DAE dominates delegatee's DAE must be removed from delegatee. That is, delegatee can still have those permissions their DAEs the delegatee's DAE still dominates.

It seemed that this mode can make delegation and revocation more flexible. But indeed it has some shortcomings or security risks:

1. Partial revocation means the delegation model must decide which permission can be removed from delegatee. That is, if a delegatee's DAE still dominates a delegated permission's DAE, the delegatee can keep this delegated permission and a delegated permission must be removed from the delegatee if the delegated permission's DAE dominates the delegatee's DAE. It will increase the computational complexity of revocation, especially in an environment with frequent delegation and revocation.

2. A temporary delegation role's permissions always are the least privileges that are actually needed by a task and these permissions must be treated as a whole in delegation and revocation. Partial revocation means these permissions can be divided into different groups. So, it has broken the integrality of delegation permissions. Also, it's difficult to implement.

The extended ABDM supports total automatic revocation.

IV. System Implementation

A. Auto Revocation Algorithm

In existing delegation models, revocation always performed when a session has deactivated. It has reduced system overload, but it is unsafe in some situations, especially in a muti-step delegation.

In the situation (1) of fig. 2, supposing u own delegation role r during the period of [a, c] and u deactivate session in which r is active at time b, r will not revoked from u for time b is prior to time c. So, if u active r at d, the operation is illegal for u cannot own r at time d. Similarly, if u.DAE cannot satisfy r.DAE at time c, the operation is also illegal if u active r at time d.

In the situation (2) of fig. 2, if u deactivates session at time b and u wants to delegate r to other person, the operation will be illegal for u will not own r after time c. Similarly, if u.DAE cannot satisfy r.DAE at time c, the operation is also illegal if u delegates r to other person at time d. A solution is to check whether a user satisfy a delegation's CR or DAE frequently. But, if the interval between two time checkpoints, at which the system will judge whether there are some user role assignment must be revoked automatically, is two small, it will reduce the system performance sharply, especially in a situation where there exists huge quantity of user role assignments. Because a revocation must be performed when a user does not activate the role which must be revoked from the user, auto revocation will be performed before a session activation and deactivation. That is, auto revocation triggered by time will not performed at each time checkpoint.

Here we give a delegation relation list in table I, which contains four main fieleds: delegation role, delegatee, delegation duration and CR. A delegation relation will be inserted or deleted from the list by the system if a delegation has performed or revoked:

Here we give an auto revocation algorithm as follows:

Algorithm: auto-revocation

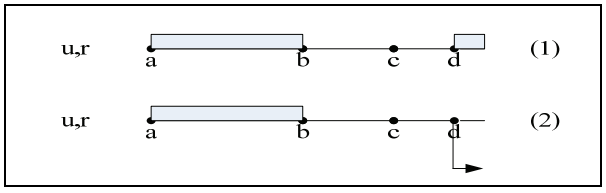

Figure 2. Example of active and delegate a delegate role

TABLE I.
DELEGATION RELATION LIST

Delegation role	delegatee	Delegation duration	CR
r1	u2	[2008/1/1 0:0:0, 2008/2/1 12:0:0]	r5,r7
r1	u100	[2008/6/3 11:0:0, 2008/6/4 10:0:0]	r11
r10	u201	[2009/10/1 0:0:0, 2009/10/7 23:59:59]	r5,r7,r9
r5	u34	[2008/12/2 7:0:0, 9999/12/31 23:59:59]	r20,r21
r20	u201	[2009/1/26 0:0:0, 2009/1/31 23:59:59]	r1

Input: u, R_u, LU_{DAE}, LU_{CR}, LR_{DAE}, L_d are sets of users, u's roles, list of user which DAE has changed, list of user which CR has changed, list of role which DAE has changed and delegation list respectively.

Output: modified LU_{DAE}', LU_{CR}', LR_{DAE}' and L_d'.

Steps:

1. For i=1 to $|R_u|$ do
2. If u, r_i in L_d then
3. Get delduration(u,r_i) from L_d
4. If ↓delduration(u,r_i)=true then goto 12; //triggered by time
5. If r_i in LR_{DAE} then goto 7 ; //judge whether r_i.DAE has changed
6. If u not in LU_{DAE} then goto 8; //judge whether u.DAE has changed
7. If ¬u.DAE ▷ r.DAE then goto 12;
8. If u ∉ LU_{CR} then goto 15
9. For $r_k \in cr(r_i)$ do
10. if r_k not in roles(u) then goto 12
11. Next k // triggered by CR
12. Revoke r_i from u and delete related data from L_d;
13. If u not in L_d then delete u from LU_{DAE} and LU_{CR}
14. If r_i not in L_d then delete ri from LR_{DAE}
15. ENDIF
16. NEXT i
17. Return LU_{DAE}', LU_{CR}', LR_{DAE}', L_d'

Supposing the number of delegation relations of system is n and the total number of roles is m, the time complexity of this algorithm is O(|n×m|).

B. System Architecutre

Here we illustrate the main components of user role assignment and revocation by fig. 3. In fig. 3, the solid arrow lines denote data flow between components and databases, while dashed arrow lines denote data flow among components. Components' functions can be found in table II.

We explain how those components work in these three types of auto-revocation:

- Revocation Triggered by Delegation Duration
  1. Event Monitor (EM) detects a session activation or deactivation and judges its type and sends a message to Duration Judgment

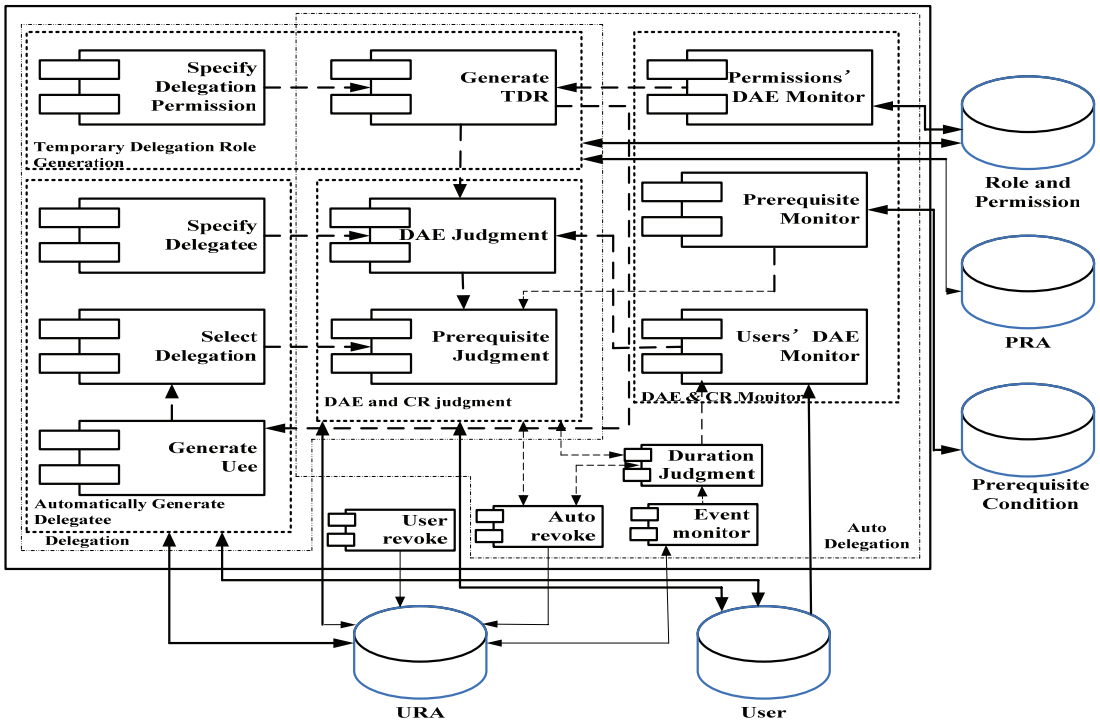

Figure 3. System Architecture

TABLE II.
FUNCTIONS OF COMPONENTS

Component	Function
User, Role and Permission, PRA, Prerequisite Condition and URA	Save user, role, permission data, related DAEs, URAs, PRAs and CR
Specify delegation permission	Specify delegation permissions set
Generate TDR	Generate a tdr and it's DAE and CR
Specify delegatee	Select delegatee in a DDD type delegation
Generate Uee	System generates delegatees satisfy delegation constraints automatically for a UDD type delegation
Select delegatee	Delegation select a delegatee from Uee
DAE seniority judgment	Judge whether a user's DAE is senior to a tdr'DAE。
User's DAE monitor	Monitor whether a user's DAE has changed and generates LU_{DAE}。
Permission's DAE monitor	Monitor whether a permission's DAE has changed and generates LR_{DAE}。
Prerequisite monitor	Monitor whether a user's CR has changed and generates LU_{CR}。
Event monitor	Monitor session activation, deactivation and further delegation event of URA and get Ld
Duration judgment	Judge whether an delegation duration is invalid
Auto revoke	Exective auto-revocation-judement alogrithm and save reults into URA
User revoke	Revoke a delegation by user

(DJ);

2. DJ decides whether this revocation can be triggered by delegation duration. If yes, it send the message to Auto revoke;
3. Auto revoke component perform an auto-revocation and saves revocation results into

URA.

- Revocation Triggered by User's Delegation Attribute Expression
  1. Event Monitor (EM) detects a session activation or deactivation and judges its type and sends a message to Duration Judgment (DJ);
  2. DJ decides whether this revocation can be triggered by delegation duration. If not, it send the message to DAE & CR Monitor (DCM);
  3. DCM decides whether the user's or permission's delegation attribute expression or CR has been changed. If the user's DAE changed, DCM sends the message to DAE and CR judgment (DCJ);
  4. DCJ decides whether the changed DAE of a user satisfies a temporary delegation role's DAE. If yes, DCJ sends the message to Auto revoke;
  5. Auto revoke component perform an auto-revocation and saves revocation results into URA.
- Revocation Triggered by Permission's Delegation Attribute Expression
  1. Event Monitor (EM) detects a session activation or deactivation and judges its type and sends a message Duration Judgment (DJ);
  2. DJ decides whether this revocation can be triggered by delegation duration. If not, it send the message to DAE & CR Monitor (DCM);
  3. DCM decides whether the user's or permission's delegation attribute expression or CR has been changed. If a permission's DAE changed, DCM sends the message to DAE and CR judgment (DCJ);
  4. DCJ sends a message containing changed permission's DAE to Temporary Delegation Role Generation (TDRG);
  5. TDRG regenerate the temporary delegation role's DAE and sends a message to DCJ;
  6. DCJ decides whether the DAE of a user satisfies the changed DAE of a temporary delegation role. If yes, DCJ sends the message to Auto revoke;
  7. Auto revoke component perform an auto-revocation and saves revocation results into URA.
- Revocation Triggered by CR
  1. Event Monitor (EM) detects a session activation or deactivation and judges its type and sends a message to Duration Judgment (DJ);
  2. DJ decides whether this revocation can be triggered by delegation duration. If not, it send the message to DAE & CR Monitor (DCM);
  3. DCM decides whether the user's or permission's delegation attribute expression or CR has been changed. If CR changed DCM sends the message to DAE and CR judgment (DCJ);
  4. DCJ decides whether the CR of a user satisfies the constraints of this delegation. If not, DCJ sends the message to Auto revoke;
  5. Auto revoke component perform an auto-revocation and saves revocation results into URA.

C. Case Study

Supposing in a software project, the project needs more testing engineer to test the software. So, the project manger can temporarily delegate p1, p2 and p3 to other engineers, say programmer, who can temporarily work as testing engineer to complete the testing task. Table III gives some engineers' DAEs. According to table III, only u1 and u3 can be assigned to role tdr. Here we elaborate the revocation operations:

- Revocation Triggered by Delegation Duration

Lets consider row 4 of table I, u201 has been delegated role r10 with the delegation duration [2009/10/1 0:0:0, 2009/10/7 23:59:59]. So if current time has past 2009/10/7 23:59:59, r10 can be revoked from u201 automatically.

- Revocation Triggered by User' Delegation Attribute Expression

In this case, the testing engineers are testing module B now. According to the rule of software engineering, the programmer who has designed module B is prohibited from testing it. Supposing mutual B needs some modifications and user u1 is now modifying it. So, u1's DAE must be changed into: familiar_test_tool≥1 AND testing_experience≥3 AND language=JAVA AND database=ORACLE AND familiar_with_test_theory=yes AND current_module=B. Obviously, user A's delegation attribute expression can not satisfied permissions p1, p2 and p2's delegation attribute expressions any more. So, these three permissions must be revoked from u1.

- Revocation Triggered by Permission's Attribute

TABLE III.
USERS AND THEIR DAES

User	DAE
u1	familiar_test_tool≥1 AND testing_experience≥3 AND language=JAVA AND database=ORACLE AND familiar_with_test_theory=yes AND current_program _module=A
u2	Language=VB AND database=ORACLE AND familiar-_with_test_theory=yes AND current_program _module =none
u3	familiar_test_tool≥1 AND testing_experience≥3 AND language=JAVA AND database=ORACLE AND current_program_module= A AND familiar_with_test_theory=yes

Expression

In this case, supposing the testing task needs all the testing engineers must be accomplished in SQL SERVER. So the delegation attribute expressions of permission p1, p2 and p3 must be changed into: language=JAVA AND testing_experience ≥2 AND database=SQL SERVER AND familiar_with_test_theory=yes AND current_module≠B 、 familiar_test_tool≥1 AND language=JAVA AND database=SQL SERVER AND current_module≠B 、 familiar_test_tool≥1 AND testing_experience≥2 AND language=JAVA AND database=SQL SERVER AND familiar_with_test_theory=yes AND current_module≠B respectively. Accordingly, if user u1 and u3 are not accomplished in SQL SERVER, their delegation attribute expressions do not satisfy permission p1, p2 and p3's delegation attribute expressions, and these three permissions must be revoked form them.

- Revocation Triggered by CR

Let's consider examples in table I again. Supposing r5 will not assign to u201 anymore, so r10 will be revoked from u201 automatically because r10 ∈ roles (u201) ∧¬u.CR▷ CR(u, r10)) ∧(CRModified(u,r10)=true).

V. CONCLUSION

As a further research results of [11], this paper focuses on auto revocation in delegation and thus makes the ABDM a complete and security delegation model. The most important type of revocation is revocation triggered by user's or delegation role's DAE. Our model takes delegation security as well as system efficiency into consideration and makes a balance between these two sides.

According to [7], delegation and revocation can be divided into many types. Our model supports followings types: temporary and permanent, monotonic and non-monotonic, total and partial, self-acted and agent-acted (UDD type of delegation), single step and multi-step delegation and related revocation. We believe auto revocation in cascaded delegation is an interesting topic for further research.

ACKNOWLEDGMENT

This research is supported by National Natural Science Foundation of China (Project No. 60803027), the Natural Science Foundation of Chongqing, China (Project No. CSTC, 2008BB2320) and the National High Technology Research and Development Program of China (Project No. 2007AA01Z445)

REFERENCES

[1] Ravi Sandhu, Edward Coyne, Hal Feinstein and Charles Younman, "Role-Based Access Control Models", *IEEE Computer* 29(2), pp. 38-47.

[2] Jacques Wainer, Akhil Kumar, "A Fine grained, Controllable, User to User Delegation Method in RBAC", in *proceedings of SACMAT'05*, June 1–3, 2005, Stockholm, Sweden.

[3] Roberto Tamassia, Danfeng Yao, and William H. Winsborough, "Role-based cascaded delegation", In *Proc of the SACMAT'04,* Yorktown Heights, New York, USA: ACM press, 2004.

[4] Barka, E.; Sandhu, R., "Framework for Agent-Based Role Delegation", *In proceedings of ICC'07(IEEE International Conference on Communications*, 24-28 June 2007, pp.1361 – 1367.

[5] Abdallah, A.E.; Takabi, H., "Integrating Delegation with the Formal Core RBAC Model", In *proceedings of Information Assurance and Security,* 8-10 Sept. 2008, pp.33 – 36.

[6] Longhua Zhang, Gail-Joon Ahn, and Bei-Tseng Chu, "A rule-based framework for role-based delegation:, In:*Proc of SACMAT'01*, Chantilly, VA, USA: ACM press, 2001.

[7] Ezedin S. Barka, "Framework for Role-Based Delegation Models" [PhD Dissertation], George Mason University, Fairfax, Virginia, summer 2002.

[8] Xinwen Zhang, Sejong Oh, and Ravi Sandhu, "PBDM: A Flexible Delegation Model in RBAC", In *Proc of the SACMAT'03,* Como, Italy: ACM press, 2003.

[9] ZHAO Qing-Song, SUN Yu-Fang, and SUN Bo, "RPRDM: A Repeated-and-Part-Role-Based Delegation Model", *Journal of Computer Research and Development*, 2003, 40(2), pp.221-227.

[10] Mohammad Abdullah Al-Kahtani, "A family of Models for Rule-Based User-Role Assignment" [PhD Dissertation], Fairfax: Department of computer science of George Mason University, 2003.

[11] ZHAI Zheng-De, "Quantified-Role Based Controllable Delegation Model", *Chinese Journal of Computers*,2006,29(8)

[12] SUN Bo , ZHAO Qing-Song , and SUN Yu-Fang, "TRDM —Temporal Role-Based Delegation Model", *Journal of Computer Research and Development*, 2004,41(7), pp.1104~1109.

[13] Jacques Wainer, Akhil Kumar, "A Fine grained, Controllable, User to User Delegation Method in RBAC", in *proceedings of SACMAT'05*, June 1–3, 2005, Stockholm, Sweden.

[14] Ye Chun-Xiao, Wu Zhong-Fu, Fu Yun-Qing, et al, "An Attribute-Based Extended Delegation Model", *Journal of Computer Research and Development*,2006,43(6), pp.1050-1057.

Chunxiao Ye was born in Chongqing, China, on April 12th, 1973. He received his BEng in computer software from Sichuan University, China in 1995 and MEng in computer architecture from Chongqing University, China in 2002. In 2005, he received his PhD in computer science from Chongqing University, China.

He is now an associate professor at the College of Computer Science, Chongqing University, China. He has published more than 50 journal papers and conference articles in access control, software engineering and database. His research interests include access control, grid, software engineering and database.

Dr. Ye is now a senior member of China Computer Society and a member of The Institution of Engineering and Technology (IET).

Xiang Li was born in Chongqing, China, in 1979. He received his BEng in computer from Chongqing University, China in 2002 and MEng in software engineering from Chongqing University, China in 2007. He is now a engineer in Chongqing Survey Institute.