Flexible and Manageable Delegation of Authority in RBAC

Tuan-Anh Nguyen, Linying Su, George Inman, David Chadwick
Computing Laboratory, University of Kent, England
{tn32, ls97, G.Inman, D.W.Chadwick}@kent.ac.uk

Abstract

One of the most challenging problems in managing large networks is the complexity of security administration. Role based access control (RBAC) has become the predominant model for advanced access control. Flexibility and manageability are important requirements for any delegation system which is one of the most important access control management mechanisms in authorisation systems This paper proposes a delegation model that satisfies these requirements.

1 Introduction

The role based access control model (RBAC) proposed in [8] has become the predominant model for advanced access control because it reduces the complexity and cost of security administration in large networked applications.

Most large organisations have some business rules that can be related to an access control policy. Delegation of authority can be seen to be one of these rules and is described as the process whereby user A authorises another user B to act on A's behalf, by sharing a set of A's permissions, possibly for a specific period of time ([6]).

There are two basic requirements for a delegation of authority model:

- flexibility: one authority should be allowed to delegate any subset of his permissions to subordinates ([9], [13]),
- manageability: authorities should have the capability to specify what their subordinates can do ([2], [13]).

In this article, we restrict our scope to user-to-user role-based delegation and we propose a delegation model that satisfies these requirements.

This paper is organized as follows. In the next section we will present our model for delegation of authority. In section 3, we will discuss revocation of authority and the final sections will discuss both the related work and our conclusions.

2 Delegation of Authority

In order for a person to be able to perform a delegation of a role, he must have the *right to delegate the role*. In our work, we separate two concepts: *a role* and *the right to delegate a role*. A right to delegate a role is not included in the permissions associated with the role.

For each role R, we define a new associated role R^-. The permissions given to these roles are the same but a holder of the role R can assert those permissions. However, the holder of the role R^- is NOT allowed to assert the permissions given by the role R. Role R^- is similar to the noAssertion extension in an X.509 Attribute Certificate presented in [1].

Our model supports two methods of delegating roles to users. The first is **direct role delegation** where a delegator explicitly specifies a *unique delegatee identifier* in a delegation. Direct role delegation requires that a delegator to manually assign a role to a delegatee based on a unique delegatee identifier. There are cases where manual delegation is insufficient, for example, when the number of users in a group is large, manual delegations would be both impractical and infeasible. Instead, role delegation can be performed using some *expressions of delegatee attributes* rather than a unique delegatee identifier. This method of delegating roles to users through some logical expression of user attributes is known as **attribute-based role delegation** ([10]).

Furthermore, a delegator may want to restrict the way in which his delegatee can further delegate the role. Our model allows delegation with **restrictions**. The restriction of a delegation affects all of the delegations in the delegation chain, so that the delegatee is not able to freely delegate their role to anyone. Our model also supports the use of **generic constrains** that cannot be violated in any delegation.

2.1 Right to Delegate

In order to delegate a role, the delegator has to have the right to delegate the role. We define R^* as either the role R or the role R^- and a delegation right (or "right to delegate") of the role R^* in this model is represented as:

$$d(R^*, Q, n, DT)$$

where Q is a restriction of the delegation right – the holder of the delegation right can only delegate the role R^* to a user that satisfies the restriction Q. $n > 0$ is the maximum delegation depth of a delegation chain that can be made by the holder. A special case is when $n = 0$ and we denote this as $d(R^*)$. This means that the holder of the right $d(R^*)$ is not allowed to delegate the role R^* any further. DT is the *maximum validity period* of the delegations that can be made by the holder.

In general if a user holds $d(R^*, Q, n, DT)$ then he can start a delegation chain of the role R^* at most n steps long, provided that each delegatee in the delegation chain satisfies the restriction Q and does not violate any generic constraint. There is an extreme case where $n = \infty$ and we denote the delegation right for this case as $d^*(R^*, Q, DT)$. $d^*(R^*, Q, DT)$ allows a user to delegate the right $d^*(R^*, Q, DT)$ itself to another user.

2.2 Delegation Statement

A delegation right and a delegated role are given to a delegatee via a delegation statement. In our model, a delegation statement (or "delegation" for short) has the following form:

$$\mathsf{delegate}(UID, UAI, R^*, n, T, DT, Q)$$

UID specifies the unique delegator identifier – the issuer or delegator of the delegation, UAI is either an expression of the delegatee's attributes or a unique delegatee identifier. R^* is the delegated role. n is the maximum delegation depth of a delegation chain that the delegatee is allowed to make based on this delegation. T is the *validity period* of the delegation, which guarantees that the delegation will only last for a specific period of time. DT is the maximum validity period of the delegations that can be made by the delegatee based on this delegation. The separation between T and DT is needed in order to allow a user to delegate a role to another user with a longer validity period than the validity period of the user's role. For example the President of the United States Mr Bill Clinton can delegate (appoint) Mr Alan Greenspan as the Manager of The US Federal Reserve for a term of ten years even though his own office term is no longer than four years. The right to delegate and the right to assert the role are valid at any time instance t_0 where $t_0 \in T$. In this work, both the validity periods T and DT start at the same time because the removal of this condition means that the delegatee of a delegation could receive the delegated role before the delegator. The start time of T and DT is also the issuance time of the delegation. Q is the restriction on the delegatee of the delegation, e.g. in order to be a delegatee of the delegation, a user has to satisfy Q. Further discussion of UAI and Q will be provided later in Section 2.11. The delegatee of the delegation $\mathsf{delegate}(UID, UAI, R^*, n, T, DT, Q)$ receives $d(R^*, Q, n, DT)$ and the right to assert the permissions given by the role R^*, valid at $t_0 \in T$.

2.3 Comparing Restrictions - Stronger Relation

Definition 1. *If P and Q are two restrictions, we say $Q \rightarrow P$ or Q is stronger than P if the restriction P is stricter than or equal to the restriction Q.*

Here we call the less restrictive restriction stronger than more restrictive restrictions. A less restrictive restriction allows a delegator to delegate a role to more people than a more restrictive restriction. Especially, "no restriction" is stronger than any other restriction: $\emptyset \rightarrow Q$ for every Q.

2.4 Comparing Roles - Stronger Relation

Definition 2. *We say either $\mathsf{imply}(A, B)$ or role A inherits role B if all the permissions of B are also permissions of A.*

Definition 3. *The stronger relation ($\rightarrow$) among roles is defined as $A \rightarrow B$ if and only if $A = B$ (two roles are the same) or $\mathsf{imply}(A, B)$.*

2.5 Comparing Validity Periods - Stronger Relation

Definition 4. *We say that $DT1 \rightarrow DT2$ if the validity period $DT1$ is equal to or contains the validity period $DT2$.*

2.6 Comparing Delegation Rights - Stronger Relation

Definition 5. *The stronger relation ($\rightarrow$) among the delegation rights is defined as follows:*

- *If $Q \rightarrow P$ then $d(R^*, Q, n, DT) \rightarrow d(R^*, P, n, DT)$ with $n > 0$*
- *If $Q \rightarrow P$ then $d^*(R^*, Q, DT) \rightarrow d^*(R^*, P, DT)$*
- *$d^*(R^*, Q, DT) \rightarrow d(R^*, Q, n, DT) \rightarrow d(R^*, Q, k, DT)$ with $n > k > 0$*
- *If $R \rightarrow S$ then $d(R^*, Q, n, DT) \rightarrow d(S^*, Q, n, DT)$ with $n > 0$*
- *If $R \rightarrow S$ then $d^*(R^*, Q, DT) \rightarrow d^*(S^*, Q, DT)$*

- *If* $DT1 \rightarrow DT2$ *then* $d(R^*, Q, n, DT1) \rightarrow d(R^*, Q, n, DT2)$ *with* $n > 0$
- *If* $DT1 \rightarrow DT2$ *then* $d^*(R^*, Q, DT1) \rightarrow d^*(R^*, Q, DT2)$

2.7 Generic Constraints

Generic constraints are set by a system administrator in order to lay out higher-level organisational policies. No delegation should be able to violate any of these generic constraints. We express a generic constraint as:

$$\perp \leftarrow \mathsf{assign}(U, R^*, T, DT), \gamma$$

where γ is a conjunction of RBAC primitive predicates and any other derived predicates that refer to users, roles, permissions or time as stated in [11]. $\mathsf{assign}(U, R^*, T, DT)$ denotes the assignment of role R^* to user U with the validity periods of the delegation are T and DT. The constraint states that it is a violation for both $\mathsf{assign}(U, R^*, T, DT)$ and γ to be true at the same time.

2.8 Strict Delegation Acceptance

We first define the relation (predicate) $\mathsf{has}(U, d)$ (having a right to delegate) as:

Definition 6. $\mathsf{has}(U, d)$ *at a time instance* t_0 *is true if and only if: 1) there was an accepted delegation which gave* d *to* U *and* d *is valid at* t_0 *or 2) there was an accepted delegation which gave* d' *to* U*,* $d' \rightarrow d$ *and* d' *is valid at* t_0 *or 3)* U *holds* d *initially (*U *is either a system administrator or a user trusted by an organisation to have* d*) at* t_0 *or 4)* U *holds* d' *initially (*U *is either a system administrator or a user trusted by an organisation to have* d'*) at* t_0 *and* $d' \rightarrow d$*.*

Definition 7. *The* ***decrement function*** *on a delegation right is defined as follows:*

- *decrement(*$d(R^*, Q, n, DT)$*)=*$d(R^*, Q, n-1, DT)$ *with* $n > 1$
- *decrement(*$d(R^*, Q, 1, DT)$*)=*$d(R^*)$
- *decrement(*$d^*(R^*, Q, DT)$*)=*$d^*(R^*, Q, DT)$

Definition 8. *If* UAI *is a unique delegatee identifier then the delegation*

$$\mathsf{delegate}(UID, U, R^*, n, T, DT, Q)$$

at a time instance t_0 *is accepted if and only if: 1)* $\mathsf{has}(UID, d(R^*, Q, n+1, DT))$ *at* t_0*, 2)* $\mathsf{has}(UID, d(R^*, Q, n+1, T))$ *at* t_0*, 3)* T *and* DT *start at* t_0*, 4)* $Q(U)$ *is true at* t_0*, 5)* $\mathsf{assign}(U, R^*, T, DT)$ *does not violate any generic constraint and 6) for all roles* S *such that* $\mathsf{imply}(R, S)$*,* $\mathsf{assign}(U, S^*, T, DT)$ *does not violate any generic constraint.*

We call direct role delegation a **physical delegation**.

Definition 9. *If* UAI *is an expression of delegatee attributes then the delegation*

$$\mathsf{delegate}(UID, UAI, R^*, n, T, DT, Q)$$

at a time instance t_0 *is accepted if and only if: 1)* $\mathsf{has}(UID, d(R^*, Q, n+1, DT))$ *at* t_0*, 2)* $\mathsf{has}(UID, d(R^*, Q, n+1, T))$ *at* t_0 *and 3)* T *and* DT *start at* t_0*.*

To be a delegatee of this delegation, the delegatee has to satisfy all the restrictions specified in the delegation and the delegation does not violate any generic constraint. Formally:

Definition 10. *User* U *can be a delegatee of a delegation*

$$\mathsf{delegate}(UID, UAI, R^*, n, T, DT, Q)$$

at a time instance t_0 *in which* UAI *is an expression of delegatee attributes, if and only if: 1) the delegation* $\mathsf{delegate}(UID, UAI, R^*, n, T, DT, Q)$ *was accepted and* $t_0 \in T$*, 2)* $UAI(U)$ *is true at* t_0*, 3)* $Q(U)$ *is true at* t_0*, 4)* $\mathsf{assign}(U, R^*, T, DT)$ *does not violate any generic constraint and 5) for all roles* S *such that* $\mathsf{imply}(R, S)$*,* $\mathsf{assign}(U, S^*, T, DT)$ *does not violate any generic constraint.*

If user U can be a delegatee of a delegation of this type then logically, we can say the delegation

$$\mathsf{delegate}(UID, U, R^*, n, T, DT, Q)$$

is accepted at t_0. We call this delegation a **logical delegation**.

2.9 Constrained Delegation Acceptance

The strict delegation acceptance approach requires that the delegator knows exactly what roles and rights he has or what he can do in an organisation. This requirement is not always satisfied in a dynamic environment: as an organisation's policy or requirements may be changed e.g. people often change their positions, and can join or leave different working groups so their roles may change often. When a delegator tries to delegate a role (and a delegation right) which is stronger than his own role and violates his authority or the delegator does not know exactly what he can delegate to other people, constrained delegation acceptance is needed. In this case, requested delegation is "constrained" so that the delegator would not violate his authority.

Firstly, we define the concept of "intersection between validity periods". With a and b as two time instances, we denote $max(a, b)$ as a function that finds the later time instance between the two time instances a and b. We also denote $min(a, b)$ as a function that finds the earlier time instance of the two time instances a and b.

Definition 11. *A validity period T is presented as two time instances nb and na and the validity period $T1$ is presented by the two time instances $nb1$ and $na1$. The intersection between the two validity periods T and $T1$ is a validity period, denoted as $T \cap T1$ and is presented as two time instances $max(nb, nb1)$ and $min(na, na1)$. If $max(nb, nb1) > min(na, na1)$, then we say that $T \cap T1$ is an "empty" validity period and denote it as ET.*

Definition 12. *An "Empty" role is a role without any permissions, denoted as ER.*

In order to find the constrained role from the role that the delegator wants to delegate to the delegatee and the role in the delegator's delegation right, we need to find the set of common subordinate roles of these two roles in the organisation role hierarchy. If the set is empty (or there is only the ET role) then the delegation can not be constrained and is rejected. If the set is not empty, we find the role that is "closest" to the requested delegation role.

Definition 13. *The intersection of two roles R^* and S^*, denoted as $R^* \cap S^*$ is either the ER role if R^* and S^* do not have any common subordinate role in the organisation role hierarchy or a set of common subordinate roles.*

Definition 14. *A best role in a non-empty set of roles is a role for which no other role in the set is stronger than it.*

Proposition 1. *We always can find at least one best role in a non-empty set of roles.*

The proof for above proposition is straightforward.

We denote $\mathsf{best}(R^* \cap S^*)$ as the chosen role from the set (non empty) of roles $R^* \cap S^*$.

In order to be a delegatee of a delegation, the delegatee has to satisfy both the restriction in the requested delegation and the restriction in the delegator's delegation right. Therefore, the delegatee has to satisfy a restriction which is the "intersection" between the two restrictions. Formally:

Definition 15. *The intersection of two restrictions Q and $Q1$ is a restriction, denoted as $Q \cap Q1$. User U satisfies the restriction $Q \cap Q1$ or $(Q \cap Q1)(U)$ is true if and only if both $Q(U)$ and $Q1(U)$ are true.*

We denote $sorter(n, k)$ as a function that finds the smaller value in a set of two integer values n and k.

We now return to constrained delegation acceptance. If the delegator does not have any valid right to delegate at the time of issuing then obviously, the delegation is rejected. If the delegator has a set of valid delegation rights at the time of issuing, then he (or a delegation agent proposed in [4]) can choose one right to support the requested delegation. We assume that the delegator chooses to delegate a right:

$$d(S^*, Q1, k, DT1)$$

which is valid at t_0. This right to delegate is weaker than the right to delegate $d(R^*, Q, n+1, DT)$). Obviously we have:

$$d(S^*, Q1, k, DT1) \rightarrow$$
$$d(\mathsf{best}(R^* \cap S^*), Q \cap Q1, sorter(k, n+1), DT \cap DT1)$$

and the delegation can be constrained to

$$\mathsf{delegate}(UID, UAI, \mathsf{best}(R^* \cap S^*), sorter(n+1, k), T, DT2, Q \cap Q1)$$

with $DT \cap DT1 \rightarrow T$, $DT \cap DT1 \rightarrow DT2$.

If $sorter(k, n+1) = 0$ or $R^* \cap S^* = ER$ or $DT \cap DT1 = ET$ then the delegation is rejected. Otherwise, the "constrained" delegation is accepted.

2.10 Chain of Delegations and Chain of Delegation Rights

In order to explain how a user got a role and a delegation right, we define the concept of a *chain of delegations* and a *chain of delegation rights*.

Definition 16. *A chain of delegations (or "delegation chain") for a role S^* is a sequence of delegations $(del_0, del_1, \ldots, del_n)$ where each del_i is either a physical or logical delegation and del_i has the form*

$$\mathsf{delegate}(UID_i, U_i, R_i^*, n_i, T_i, DT_i, Q_i)$$

such that:

- *$R_i^* \rightarrow R_{i+1}^* \rightarrow S^*$, the delegator's role must be stronger than the delegated role and the delegated role must be stronger than the role S,*
- *UID_0 is either the system administrator or a user trusted by the organisation that initially has the right to delegate $d_0(R^*, Q_0, n_0, DT_0)$,*
- *there is no $UID_i (i > 0)$ in the delegation chain that is either the system administrator or a user trusted by the organisation that initially has the right to delegate $d_i(R_i^*, Q_i, n_i, DT_i)$,*
- *$U_i = UID_{i+1}$, that is the delegatee of del_i is the delegator of del_{i+1},*
- *decrement$(d_i(R_i^*, Q_i, n_i, DT_i)) \rightarrow d_{i+1}(R_{i+1}^*, Q_{i+1}, n_{i+1}, DT_{i+1})$, that is the delegation right delegated in del_{i+1} is at most as strong as decrement$(d_i(R_i^*, Q_i, n_i, DT_i))$,*
- *$DT_i \rightarrow T_{i+1}$, $DT_i \rightarrow DT_{i+1}$, that is the validity periods of the del_{i+1} must be weaker than the maximum validity period of the del_i,*

- *$Q_i \to Q_{i+1}$, that is the restrictions must be getting more restrictive along the chain from del_0 to del_n,*
- *there is no pair $del_i, del_j (i \neq j)$ in the delegation chain, such that $UID_i = U_j$ or there is no loop of delegation in the chain.*

Definition 17. *Associated with each delegation chain $(del_0, del_1, \ldots, del_n)$, we have a chain of delegation rights $(d_0, d_1, \ldots, d_n)$ in which, d_i is given by del_i, $\quad i = \overline{1, n}$.*

Given the delegation chain $(del_0, del_1, \ldots, del_{n-1}, del_n)$, we say that $(del_0, del_1, \ldots, del_{n-1})$ is a supporting chain for del_n. We also say that the chain of delegation rights $(d_0, d_1, \ldots, d_{n-1})$ is a supporting chain for d_n.

2.11 Discussion about Q and UAI Parameters

In this section we consider only the case in which, UAI is an expression of delegatee attributes. Because delegatee attributes are the delegatee's properties in his organisation, the expression of delegatee's attributes will vary between organisations and is application-dependent. Delegatee attributes may be the roles of the delegatee in the organisation, the delegatee's age, credit, the domain of the delegatee etc. An example expression of delegatee's attributes is

$$\bigwedge \big(\mathsf{has}(\text{Researcher role}), \mathsf{age}(35)\big)$$

in which, $\mathsf{has}(\text{Researcher role})$ means that a delegatee has a "Researcher" role and $\mathsf{age}(35)$ indicates that the age of delegatee is 35. The symbol $\bigwedge$ means that in order to be a delegatee, the user has to have the Researcher role and the user's age must be 35.

Q is the restriction posed on the delegatee of a delegation. Q can also be expressed using an expression of delegatee attributes like the UAI parameter. On a general level, $Q \in \mathbb{Q}$, in which $\mathbb{Q}$ is a set of all restrictions on user's attributes. Because both users' attributes and the restrictions on these attributes are application-dependent, it is up to an application/organisation to define a stricter relation between each element in $\mathbb{Q}$. From the stricter relation, we have the stronger relation between elements in $\mathbb{Q}$.

3 Revocation

Revocation is the process by which a delegation that was accepted is removed or retracted.

3.1 Right to Revoke

In our model, revocation is an "automatic" right for anyone who has made a delegation. For administrative purposes however it is important that others may have the right to revoke someone else's delegation. A manager can revoke any delegation that he has issued or was capable of issuing, i.e, a peer manager of the actual manager that issued a delegation can revoke it. This allows a manager to revoke a delegation issued by a colleague. Thus, if a user has a right to delegate allowing him to issue a delegation then he can revoke that delegation. Formally, whosoever has the right to delegate

$$d(R^*, Q, n, DT)$$

at t_0 can revoke a delegation

$$\mathsf{delegate}(UID, UAI, S^*, k, T, DT1, P)$$

at t_0 if $R^* \to S^*$, $Q \to P$, $n > k$, $DT \to T$ and $DT \to DT1$

3.2 Revocation Process

We define the "dependency" between delegations and delegation rights as:

Definition 18. *A delegation del_j is dependent on a delegation del_i if all supporting chains for del_j contain del_i. A delegation right d_j is dependent on a delegation right d_i if all supporting chains for d_j contain d_i.*

We classify two kinds of revocation: **non-cascading revocation** and **cascading revocation**. Given a delegation of $\mathsf{delegate}(UID, UAI, R^*, n, T, DT, Q)$, a non-cascading revocation is a revocation that removes the delegated role R^* and the delegation right $d(R^*, Q, n, DT)$ from a delegatee and the revocation takes effect only from the time that the revocation happens and it does not affect any existing delegation that is dependent on the revoked delegation. Cascading revocation is a revocation that removes the delegated role R^*, the delegation right $d(R^*, Q, n, DT)$ from the delegatee and also all of the delegations that are dependent on $d(R^*, Q, n, DT)$.

A revocation statement has the following format:

$$\mathsf{revoke}(UID, del, S, t)$$

in which, the UID is the requester of the revocation, del is the requested delegation, S is a flag indicating whether the requested revocation is a cascading or non-cascading revocation and t is the issuance time of the revocation. t is also the time that the revocation takes effect. After t, the holder of del can not assert the delegated role and can not delegate the delegated role further.

Definition 19. *The effect of $r(del_i)$, the cascading revocation of the delegation del_i, given the set of accepted delegation $\mathcal{A}$, is a new set of accepted delegations:*

$$\mathcal{A}' = \mathcal{A} - \{del_i\} - \{del_j | del_j \textit{ is dependent on } d_i\}$$

in which, the delegation right d_i is given by del_i.

The central issue in cascading revocation is to undo the cascading effects of a chain of delegations (or a chain of delegation rights).

Proposition 2. *Cascading revocation will leave no one without supporting roles.*

The proof for this proposition is straightforward.

4 Related Works

The weakness of the framework and model proposed in [3] and [4] is that they do not support multi-step delegation, constraints in delegation or role hierarchies. The SPKI model ([7]) lacks the capability to specify delegation depth and does not have any constraints on issuing name and authorisation certificates ([5]). The role-based delegation model of L. Zhang, et al. ([14], [15]) is very interesting and useful but does not support constrained delegation acceptance. The model proposed in [11] does not support attribute-based delegation and lacks time-restrictions in delegation. The RT ([10]) does not support a delegation depth constraint and allows any authority to delegate roles to anyone. From an organisation's point of view, the RT framework is very flexible but not very manageable. The delegation model in [12] supports attribute-based delegation but lacks the capability to specify generic constraints, validity periods or constrained delegation acceptance, makes the model less manageable.

5 Conclusion

In our model, the delegated role could be any role that is the same or subordinate to the delegator's role. The model has the capability of constraining requested delegations so that delegators can easily delegate roles and delegation rights to subordinates without worrying about overstepping their authority. Furthermore, the model supports direct role delegation and attribute-based role delegation, which is useful when the number of users in organisation is large.

In our model, a delegator can specify the maximum delegation depth that a delegatee can further delegate a delegated role. A delegator can delegate a role that is subordinate to his role so that the delegated role is appropriate to the delegatee's position in the organisation.By specifying a restriction in delegation statements, a delegator can control who gets the role based on his delegation. In addition, every delegation can not violate the organisation's security requirements, specified by the generic constraints.

References

[1] Itu-t recommendation x.509, iso/iec 9594-8, information technology. open systems interconnection. public-key and attribute certificate frameworks, 2005.

[2] J. Bacon, K. Moody, and W. Yao. A model of oasis role-based access control and its support for active security. *ACM Transactions on Information and System Security*, 5(4):492–540, 2002.

[3] E. Barka and R. Sandhu. A role-based delegation model and some extensions. In *In Proceedings of the 23th National Information Systems Security Conference (NISSC 2000), Baltimore, USA*, 2000.

[4] E. S. Barka and R. Sandhu. Framework for role-based delegation models. In *16th Annual Computer Security Applications Conference (ACSAC'00)*, pages 168–177, 2000.

[5] O. Canovas and A. F. Gomez. A distributed credential management system for spki-based delegation systems. In *Proceedings of 1st Annual PKI Research Workshop, Gaithersburg, Maryland, USA*, 2002.

[6] B. Crispo. Delegation protocols for electronic commerce. In *Sixth IEEE Symposium on Computers and Communications (ISCC'01) 07 03-07 2001 Hammamet, Tunisia*.

[7] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. Rfc 2693 - spki certificate theory, September 1999.

[8] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.

[9] L. Kagal, T. Finin, and Y. Peng. A delegation based model for distributed trust. In *Proceedings of the IJCAI01 Workshop on Autonomy, Delegation and Control: Interacting with Autonomous Agent, Seattle*, pages 73–80, 2001.

[10] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, 2002.

[11] J. Wainer and A. Kumar. A fine-grained, controllable, user-to-user delegation method in rbac. In *SACMAT'05*, pages 59–66, Stockholm, Sweden, 2005. ACM.

[12] C. Ye, Z. Wu, and Y. Fu. An attribute-based delegation model and its extension. *Journal of Research and Practice in Information Technology*, 38(1):3–17, February 2006.

[13] G. Yin, H.-m. Wang, D.-x. Shi, Y. Jia, and M. Teng. A rule-based framework for role-based constrained delegation. *ACM International Conference Proceeding Series, Proceedings of the 3rd international conference on Information security, Shanghai, China*, 85:186 – 191, 2004.

[14] L. Zhang, G. J. Ahn, and B. T. Chu. A rule-based framework for role-based delegation. *ACM SACMAT'01 Chantilly, Virginia, USA*, 2001.

[15] L. Zhang, G. J. Ahn, and B. T. Chu. A rule-based framework for role-based delegation and revocation. *ACM Transactions on Information and System Security*, 6(3):404–441, 2003.