

Modeling Role-based Trust and Authority in Open Environments

Dongwan Shin

Department of Computer Science
New Mexico Tech
Socorro, NM 87801
E-mail: doshin@nmt.edu

Abstract

Trust and authority are essential components that must be factored in designing and implementing a distributed access control mechanism for open environments. This paper presents a role-based approach to modeling trust and authority for open and distributed computing environments. Specifically, our approach centers on the extension of role-based capability delegation from local domains to trusted domains. A formal specification based on the fixpoint semantics is presented in order to better understand and design a distributed access control mechanism that can be built on our approach.

Keyword: Distributed access control, role-based authorization, trust management.

1. Introduction

The paradigm shift to open and distributed computing environments has not only brought the importance of their security to the fore, but also called for a new subset of security requirements. Particularly in the area of access control, functional requirements such as 1) how to handle an access request from a principal previously unknown, 2) how to enforce distributed access control across domains, and 3) how to reconcile conflicting authority policies in distributed environments, belong to such a subset, having drawn immediate attentions from security communities [4, 14, 22].

Traditional identity-based access control approaches are mostly incapable of satisfying those functional requirements because they assume that a user should be known *a priori* when an access request is made by the user; only after authenticating the user properly, they regulate access based on the user identity or some attributes of the user such as groups, security clearances, and roles. As part of the solution to address this kind of incapability, the notion of trust has been used as an essential add-on component that must be factored in designing and implementing

an access control mechanism for open, distributed environments [3, 5, 7, 10, 13]. Trust management (TM) is one of such access control mechanisms, attempting to fulfilling those security functional requirements, especially 1) and 2), for distributed environments [4]. Central to understand the concept of TM is capability delegation, which is incorporated into a credential chain from a resource owner to an access requester; access is granted if a chain of credentials proves that a requested action complies with local access control policies of the resource owners. Hence, it provides an authorization framework for dealing with unknown users, whereby it is possible to express and evaluate distributed access control policies in open environments.

In the meantime, the issue of access control policy, particularly reconciling conflicting authority policies across domains, has been studied separately from the TM-based mechanism in the literature. This is because the approach to separating mechanism from policy has been considered to be able to facilitate a better access control solution in terms of flexibility and scalability. In addition, the issue of access control policy is generally domain-specific, derived from security requirement analysis within a specific domain. For instance, mandatory access control (MAC) policies based on security labels are usually preferred in the domain where tighter authority control is needed and the confidentiality of resources managed is a primary concern, whereas discretionary access control (DAC) policies are preferred in the domain where more flexible authority control is necessary such as assigning access rights based on need-to-know rules.

As an attempt to address the reconciliation of conflicting authority policies, using the concept of roles has been investigated in order to formulate flexible authority policies [18, 20, 21]. Roles in role-based access control (RBAC) generally represent a set of competence and responsibility pairs [9], and their advanced notions include their hierarchical structure, which represents a line of authorities within an organization [17]. Further, considering their indirect and bilateral characteristics, roles can be a convenient construct for expressing authority policies in a trust-enabled distributed access control mechanism; they provide an indirection mechanism for relating users with permissions, thereby reducing complexity and potential errors in pol-

icy management. In addition, roles can represent abilities and groups of trusted users in cross-domain environments, which often derive from bilateral agreements between organizations.

While there has been continuing research on role-based capability delegation in the literature, most of its discussions have been limited to the context of either easing administrative burdens or backup of some job functions that need to be maintained within a single domain [24, 25]. Therefore, it would be beneficial to further the discussion into the context of open and distributed environments, and especially, it would be desirable to formulate activities concerning the correlation between the notion of trust and delegated authority for the purpose of providing a reference model for designing and implementing a distributed access control mechanism. Our primary objective in this paper is to present a role-based approach to modeling trust and authority for open and distributed computing environments. Specifically, our approach centers on the extension of role-based capability delegation from local domains to trusted domains. A formal specification based on the fixpoint semantics is presented in order to better understand and design a distributed access control mechanism that can be built on our approach.

The rest of this paper is organized as follows. Section 2 shows background technologies and previous research related to our work. Section 3 presents our modeling approach, followed by Section 4 which describes a formal specification allowing us to instantiate our modeling approach. Section 5 concludes the paper.

2. Related Works

The usage of roles for access control can be found in various TM systems. One of them is ISO/IEC’s PMI, based on global namespace, utilizing X.509 attribute certificate framework [6, 8, 11, 20]. PMI is an extension of public key infrastructure (PKI) in the light of authorization. The attribute certificate binds entities to attributes such as roles or groups. Along with attribute certificates, PMI is introduced with its four models: general model, control model, delegation model, and roles model. General and control models are required, whereas roles and delegation models are optional. The general model provides the basic entities which recur in other models. On the other hand, SPKI/SDSI [7, 15] is another TM system, based on local namespace, supporting the usage of roles.

Recently, the practicality of SPKI/SDSI’s capability-style approach to granting permissions has been questioned by Li et al [13]. They argued that capability-style TM systems generally lack the expressive power for role-based or group-based authorization policies, and thus they frequently result in heavy administrative burdens on public key management and distribution. As an alternative, they proposed RT framework, a family of role-based TM languages which are able to express policies and credentials for distributed authorizations. Although the RT framework

provides the rich expressive power for authorization policies and authority delegation, it still needs to address some important issues. One of them is that the distinction between a user and a principal (generally public key) is not clear in their framework, and thus it seems to be difficult to express some constraints that should be applied to the user.

3. Modeling Approach

In this section, we discuss a role-based approach to modeling trust and authority. Specifically, we base our approach on an existing RBAC model [17]. A component called trust assignment (TA) that can correlate with the notion of trust and role-based authority delegation is presented as an add-on to the model.

3.1. Trust and Authority Modeling

At the heart of our approach to modeling trust and authority is a new component called trust assignment (TA) [19]. Similar to assignment relations discussed above, trust assignment (TA) can be described as a trust-to-role assignment relation. However, the relation does not seem to be self-explanatory, since the trust *assignable* to roles is undefined yet. Hence, we 1) define what the assignable trust is, 2) identify components appropriate for representing the trust, and 3) re-define TA based 1) and 2). We first define authority domains. Each of authority domains in our model represents a single RBAC domain where RBAC administration and enforcement can take place.

Definition 1 Let D denote a set of authority domains. $D = \{d_1, \dots, d_o\}$.

The degree of trust determines the amount of capability delegable from one authority domain to another. That is, how much d_i can trust d_j can be defined as how much capability to do some tasks within d_i is given to d_j by d_i . Since roles generally represent capability, we believe that roles and their hierarchical structure are a good means to specify the degree of trust; since roles are hierarchically organized (mathematically partially ordered), their hierarchy can be naturally viewed as such. We call these roles local roles (LR), and LR is used to represent the authority to be delegated to other authority domains. On the other hand, roles in trusted authority domains are useful in two aspects; they help avoid listing all users in the trusted authority domains that will be given the delegated authority and also facilitate cascading authority delegation. We call these trusted domain roles (TDR). Central to our modeling approach is the idea of trust-based role association between LR and TDR , which enables authority delegation from a local domain to trusted domains. This is backed by TA, as shown in Figure 1.

Definition 2 Roles R are either local roles (LR) or trusted domain roles (TDR). Suppose TDR_{d_i} denotes the subset of trusted roles from authority domain d_i in the local domain,

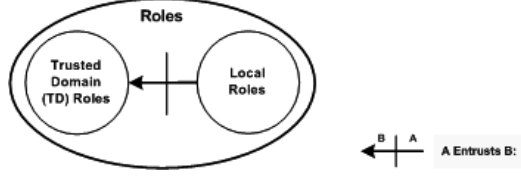


Figure 1. Authority delegation expressed by entrust operation in TA

$TDR = \bigcup TDR_{d_i}$, where $i = 1, \dots, n$. We let $R = LR \cup TDR$ denote the set of all roles.

Definition 3 Trust assignment TA is a many-to-many LR to TDR assignment relation.

We call *entrusting* a role to refer to the association of an element of LR with an element of TDR . Similarly, we call *distrusting* a role to refer to the break-up of their association. Roles can entrust or can be entrusted by other roles only across authority domains. We consider trust assignment relation to be role dominance relation but in a reverse order. For the role entrusting and distrusting operations, we use the notation $A \Rightarrow_C B$ to represent that role A entrusts role B under the condition of satisfying a set of constraints C , and $A \not\Rightarrow B$ to represent that role A distrusts role B . $A \Rightarrow_C B$ means that role B is at least as powerful as role A ; role B inherits all or partial permissions associated with role A depending upon C , and role A inherits all users assigned to role B . It should be noted that one type of constraints in C may specify the degree of authority delegation on role B . We will discuss different types of constraints in a later section.

Definition 4 Suppose we are given the relation $A \Rightarrow_C B$, we call A as an entrusting role and B as an entrusted role. Similarly, when we are given $A \not\Rightarrow B$, we call A as a distrusting role and B as a distrusted role. Local roles cannot be either an entrusted role or a distrusted role.

TA involves two types of assignment relation: *explicit* and *implicit*. Trust assignment is explicit when a role from LR entrusts a role from TDR . On the other hand, trust assignment can be made implicitly between LR and TDR_{d_j} when a role from TDR_{d_i} , which is entrusting a role from TDR_{d_j} , is entrusted by a role from LR , where domains d_i and d_j are different.

Definition 5 As we discussed earlier, trust relation is a role dominance relation, i.e., a partially ordered set. Suppose there is a trust assignment, $ta = (lr, nlr)$ where $lr \in LR$, $nlr \in TDR$, and $lr \Rightarrow_C nlr$. We say ta is explicit if and only if lr is covered by nlr , i.e., $lr \Rightarrow_C x$ and $x \Rightarrow_C nlr$ implies $lr = x$. Otherwise, ta is implicit.

3.2. Role Hierarchies

Since TA introduces another type of role dominance relation, the need for its discussion in terms of RH is apparent when we come to deal with its semantics. We call this as a *trust* aspect of role hierarchy. We also refer to this as cross domain role hierarchy ($CDRH$)¹. We believe that the inclusion of $CDRH$ into RH as distinct is unnecessary since $CDRH$ is the subset of the union of TA and RH . However, $CDRH$ deserves further discussion for its merits; it is *dynamically* constructed in an order-preserving way, as the determination of its path is dependent upon the discovery of credentials expressing TA .

In our modeling approach, role hierarchies are extended to trusted domains without a loss of their generality. Sandhu discusses two types of role hierarchies, permission inheritance RH and activation RH [16]. Permission inheritance RH concerns that a member of a senior role in the hierarchy inherits permissions from juniors, whereas activation RH means that a member of a senior role in the hierarchy is authorized to activate juniors for the purpose of least privilege. We already discussed $CDRH$ enabling permission inheritance RH . $CDRH$ can also be used for the purpose of the activation RH gracefully. However, the activation of junior roles needs to be considered only if they belong to LR . Role activation is closely related to a session, which associates a user with possibly many roles. It seems to be unreasonable that in such a session, a user from a trusted domain can activate roles from other trusted domains than local domain. Accordingly, *roles*, which is the component concerning the roles activated in a session, should be modified, while *user* remains unchanged.

Definition 6 *roles*: $S \rightarrow 2^R$ is a function mapping each session s_i to a set of roles, where $roles \subseteq \{r | (\exists r' \succeq r)[(user(s_i), r') \in UA] \wedge (r \in LR)]\}$ and session s_i has the permissions $\bigcup_{r \in roles(s_i)} \{p | (\exists r'' \preceq r)[(p, r'') \in PA]\}$.

3.3. Constraints

Continuing efforts have been made to identify and specify different types of constraints in RBAC models [2, 12]. In general, constraints previously identified in [2] can be used in our model without losing their original motivation and intention. What we are particularly interested in this section is some types of constraints that can regulate the behavior of TA , and thus they can be expressed as various security policy constructs. As we discussed earlier, TA involves authority delegation in cross-domain environments. Therefore, we strongly believe that stricter constraints should be applied to the authority delegation enabled by TA . This is why we reason about some important properties related to TA as follows.

¹Note that the relation \Rightarrow in $CDRH$ could be rewritten as \preceq for the sake of consistency in dominance relation, where all associated users and permissions are inherited. In such a case, $A \Rightarrow B$ is rewritten as $A \preceq B$.

Bilaterality - Roles from different domains are associated based on trust relations between those domains. Our assumption is that the trust relation is established by a mutual agreement between those domains, and such an agreement should be negotiated bilaterally.

Variability - Trust is subject to change with environmental conditions such as time and location. That is, any change in a trust relation should be reflected on the delegated authority of a role involved in the relation.

Granularity - Trust varies in degree. That is, the granularity of authority delegation should be determined depending on the trust degree of a trusting domain upon a trusted one.

Transitivity - Trust relations can be transitive. That is, authority delegation using roles may be cascaded through one or more trust relations among domains. The cascading authority delegation should be managed by a controllable mechanism such as a boolean or an integer approach; the boolean control simply specifies an ability to delegate, while the integer control specifies the number of possible delegation.

Based upon the properties above, we can identify some constraints relevant with TA . The variability can be expressed by *temporal* or *spacial* constraints. The temporal constraint pertains to the specification of the validity period of TA , while the spacial constraint concerns the specification of the effective domain. Similarly, TA needs to be constrained by the granularity. It is commonly believed that the level of trust is different depending on trusted entities. Granularity can be expressed as an *ordinal* constraint in trust assignment. The ordinal constraint is related to the specification of the magnitude of authority delegation in TA . On the other hand, the depth of trust is another factor having an effect on TA . Trust depth can be articulated by boolean control or integer control constraint, which we discussed above.

4. Model Instantiation

We follow the conventional approach to defining principals as entities making or authorizing access requests; principals are public keys. An authority domain is represented by a set of public keys; one or more public keys may be bound to an authority domain if they belong to the same authority domain. We typically use $K_{AttrService}$, K'_{d_i} , and K''_{d_i} for denoting specific public keys bound to authority domains. We interpret the binding between an authority domain and a public key as an assertion of a *speak-for* relation, borrowed from [1]. Similarly, one or more public keys are bound to a user if they belong to the same user.

An identifier is an arbitrarily long sequence of alphabetic and numeric characters that is used to identify the following components: roles, users, and permissions. In this work, we borrowed the concept of local namespace from SPKI/SDSI [7, 15]. Roles, users, and permissions can be expressed by a principal representing an authority domain

followed by their identifier whose typical example is their names. Based on the expressions of roles, users, and permissions, five different types of credentials can be expressed as a form of $A \rightarrow B$, which can be read as “B is at least as powerful as A.”.

4.1. Our Scheme

Our attempt to instantiate the model, described in Table 2, should be read along with Table 1, which contains a mathematical framework to understand TM systems better [23]. The framework, depicted in Table 1, consists of five elements; principals, authorizations, authorization maps, licenses, and assertions. A principal is an entity that makes or authorizes access requests. An authorization pertains to the permission granted by a principal. Authorizations are organized as a lattice, which is an important characteristic that enables multiple authorizations made by the same principal into a single authorization. An authorization map is a function which associates a principal with an authorization. A license is a monotone function (denoted by \rightarrow_m) which associates an authorization map with an authorization. The monotonicity means that authorizations can only increase by adding other principals’ authorizations in an authorization map. An assertion pertains to an authorization expression made by a principal, which can be viewed as credentials or certificates.

The lower part of Table 1 describes the semantics of assertions and TM engines to handle authorization decisions. The semantics of assertions, denoted by $\mathcal{M}_{Assertions}$, is a function which associates a set of assertions with an authorization map. Simply put, given a set of assertions made by various principals, $\mathcal{M}_{Assertions}$ returns a coherent authorization map that contains the authorization granted by each of the principals. A least fixpoint, denoted by lfp , is taken to find such an authorization map². A TM engine makes an access control decision on the basis of an authorizing principal, a request, and a set of assertions. The engine computes an authorization map from the set of assertions, and determines if the request is less than authorizations granted by the authorizing principal.

The upper part of Table 2 describes various components in our modeling approach. It also defines the authorization lattice. The lower part of the table describes the semantics of various assertions used in the scheme.

An *ident* is a function mapping principals to users that describes users identified by each principal. For instance, $i(K_A) = Alice$ where $K_A \in Principal$ and $Alice \in User$ means that a principal K_A belongs to (or identifies) a user *Alice*. We define authorizations *Auth* in our scheme as a function lattice under the pointwise ordering, where an authorization maps a principal to a set of permissions. For instance, $K_A \rightarrow \{e_1, e_2\}$ where $e_1, e_2 \in Permission$ means that the principal K_A has the set of permissions $\{e_1, e_2\}$.

² λ in Table 1 is used as a function in $A \rightarrow B$ in which $\lambda a.e$, given a where $a \in A$, returns the result of expression e where $e \in B$.

$p \in \text{Principal}$
 $u \in \text{Auth}$
 $m \in \text{AuthMap} = \text{Principal} \rightarrow \text{Auth}$
 $l \in \text{License} = \text{AuthMap} \rightarrow_m \text{Auth}$
 $a \in \text{Assertion} = \text{Principal} \times \text{License}$

$\mathcal{M}_{\text{Assertions}} : \mathcal{P}(\text{Assertion}) \rightarrow_m \text{AuthMap}$
 $\mathcal{M}_{\text{Assertions}}(A) = \text{lfp}(\lambda m. \lambda p. \sqcup \{l(m) \mid \langle p, l \rangle \in A\})$

$\mathcal{M}_{\text{Engine}} : \text{Principal} \times \text{Auth} \times \mathcal{P}(\text{Assertion}) \rightarrow \text{Bool}$
 $\mathcal{M}_{\text{Engine}}(p, u, A) = u \sqsubseteq \mathcal{M}_{\text{Assertions}}(A)(p)$

Table 1. A framework for understanding TM Systems

$s \in \text{Users}$
 $e \in \text{Permissions}$
 $r \in \text{Roles} = \text{LRole} + \text{TDRole}$
 $\langle s, r \rangle \in \text{UA} = \text{Users} \times \text{Roles}$
 $\langle p, r \rangle \in \text{PA} = \text{Permissions} \times \text{Roles}$
 $\langle r', r'' \rangle \in \text{TA} = (\text{LRole} \times \text{TDRole})$
 $\langle r', r'' \rangle \in \text{RH} = (\text{LRole} \times \text{LRole}) + (\text{TDRole} \times \text{TDRole})$

$i \in \text{Ident} = \text{Principal} \rightarrow \text{Users}$
 $u \in \text{Auth} = \text{Principal} \rightarrow \mathcal{P}(\text{Permissions})$
 $\text{UAssertion} = \text{Principal} \times \text{UA}$
 $\text{RHAssertion} = \text{Principal} \times \text{RH}$
 $\text{TAssertion} = \text{Principal} \times \text{TA}$

$\mathcal{M}_{\text{UALicense}} : \text{UA} \times \text{AuthMap} \rightarrow \text{Auth}$
 $\mathcal{M}_{\text{UALicense}}(\langle s, r \rangle, m) = \lambda p'. \{e \mid i(p') = s \text{ and } \langle e = m(p)(p'), r \rangle \in \text{PA}\}$

$\mathcal{M}_{\text{UAssertion}} : \text{UAssertion} \rightarrow \text{Assertion}$
 $\mathcal{M}_{\text{UAssertion}}(p, \langle s, r \rangle) = \langle p, \lambda m. \{\mathcal{M}_{\text{UALicense}}(\langle s, r \rangle, m)\} \rangle$

$\mathcal{M}_{\text{RHLicense}} : \text{RH} \times \text{AuthMap} \rightarrow \text{Auth}$
 $\mathcal{M}_{\text{RHLicense}}(\langle r', r'' \rangle, m) = \lambda p. \sqcup \{ \exists \langle i(p), r \rangle \in \text{UA},$
 $\text{if } (r' \preceq r), \text{ then } \mathcal{M}_{\text{UALicense}}(\langle s, r' \rangle, m)(p), \mathcal{M}_{\text{UALicense}}(\langle s, r'' \rangle, m)(p) \}$

$\mathcal{M}_{\text{RHAssertion}} : \text{RHAssertion} \rightarrow \text{Assertion}$
 $\mathcal{M}_{\text{RHAssertion}}(p, \langle r', r'' \rangle) = \langle p, \lambda m. \{\mathcal{M}_{\text{RHLicense}}(\langle r', r'' \rangle, m)\} \rangle$

$\mathcal{M}_{\text{TALicense}} : \text{TA} \times \text{AuthMap} \rightarrow \text{Auth}$
 $\mathcal{M}_{\text{TALicense}}(\langle r', r'' \rangle, m) = \lambda p. \sqcup \{ \exists \langle i(p), r \rangle \in \text{UA},$
 $\text{if } (r' \Rightarrow r), \text{ then } \mathcal{M}_{\text{UALicense}}(\langle s, r' \rangle, m)(p), \mathcal{M}_{\text{UALicense}}(\langle s, r'' \rangle, m)(p) \}$

$\mathcal{M}_{\text{TAssertion}} : \text{TAssertion} \rightarrow \text{Assertion}$
 $\mathcal{M}_{\text{TAssertion}}(p, \langle r', r'' \rangle) = \langle p, \lambda m. \{\mathcal{M}_{\text{TALicense}}(\langle r', r'' \rangle, m)\} \rangle$

Table 2. Scheme for instantiating our modeling approach

| $A \in P(\text{Assertions})$ | $\text{fp}(A)$ computation | | |
|--|----------------------------|------------|-------------|
| | HotelsRUs | TravelsRUs | AttrService |
| $(K_{\text{HotelsRUs}} \langle K_{\text{HotelsRUs}}, \text{Alice}, K_{\text{HotelsRUs}}, \text{MarketingAsst} \rangle)$ | N | N | N |
| $(K_{\text{TravelsRUs}} \langle K_{\text{HotelsRUs}}, \text{MarketingAsst}, K_{\text{TravelsRUs}}, \text{TravAgent} \rangle)$ | MarketingAsst | N | N |
| $(K_{\text{AttrService}} \langle K_{\text{TravelsRUs}}, \text{TravAgent}, K_{\text{AttrService}}, \text{BizPartners} \rangle)$ | MarketingAsst | TravAgent | N |
| | MarketingAsst | TravAgent | BizPartners |

Figure 2. An example of the result of a least fixpoint computation

For authmap m , if $\lambda K_A.\{e_1, e_2\}$ is in $m(p)$, then principal p authorizes K_A to have the set of permissions $\{e_1, e_2\}$.

We define three authorization assertion types for our scheme that can represent the abstraction of authorization certificates: $U\text{Assertion}$, $RH\text{Assertion}$, and $T\text{Assertion}$. $U\text{Assertion}$ pertains to an authorization expression related to user assignment, whereas $RH\text{Assertion}$ expresses an authorization relevant to role hierarchy. $T\text{Assertion}$ is an authorization expression concerning trust assignment. For the sake of simplicity, we consider that permission assignment is set as local policies within systems. The semantics of $U\text{Assertion}$ can be understood in light of a given principal. A set of permissions is granted to the principal by a user-role relation and an authmap. For example, $\lambda K_A.\{e_1, e_2\} \in \mathcal{M}_{U\text{ALicense}}(\langle \text{Alice}, r \rangle, m)$ where $\{e_1, e_2\} \subseteq \text{Permission}$ means that K_A belonging to Alice can have a set of permissions $\{e_1, e_2\}$ that are assigned to a role r . An $U\text{Assertion}$ can be represented by $\langle p, \langle s, r \rangle \rangle$, which illustrates that a principal p authorizes a principal p' identifying a user s to have a set of permissions E that are assigned to a role r , where $\lambda p'.E \in \mathcal{M}_{U\text{ALicense}}(\langle s, r \rangle, m)$ and $E \subseteq \text{Permission}$. Given the meaning of $U\text{Assertion}$, the semantics of role hierarchy can be taken from it when certain conditions, i.e., role dominance relation, are met. The basic idea behind this is that, if there exist some user-role relations derived from a principal and if a derived role is senior to or the same that the senior role in a role hierarchy relation, then the principal can have a set of permission which is a least upper bound of two authorizations resulting from roles in the role hierarchy relation. For example, assuming $\langle e_3, r' \rangle, \langle e_4, r'' \rangle \in PA$, $\lambda K_A.\{e_3, e_4\} \in \mathcal{M}_{RH\text{License}}(\langle r', r'' \rangle, m)$ where $r' \succeq r''$ represents that if $\langle \text{Alice}, r \rangle \in UA$ and $r' \preceq r$, K_A can have a set of permissions $\{e_3, e_4\}$ which is a least upper bound of $\{e_3\}$ and $\{e_4\}$. An $RH\text{Assertion}$ can be represented by $\langle p, \langle r', r'' \rangle \rangle$, which means that a principal p authorizes a principal p' to have such a set of permissions if role dominance conditions are met. From a similar approach for the semantics of role hierarchy, we can give the semantics of trust assignment by substituting \Rightarrow for \preceq . An $T\text{Assertion}$ is represented by $\langle p, \langle r', r'' \rangle \rangle$, which means that a principal p authorizes a principal p' to have a set of permissions if trust assignment conditions are met.

Since we defined all the necessary components, let us see how our trust management engine works. Suppose we want to know if principal p authorizes p' to perform the operation

denoted by e according to the set of assertions A . We can express access request as $u = \lambda p'.E \in \text{Auth}$, where $E = \{e\}$. Given the set of assertions A and the access request u , our trust management engine computes $\mathcal{M}_{\text{Engine}}(p, u, A)$, which is equivalent to $u \subseteq \mathcal{M}_{\text{Assertion}}(A)(p)$, which is equivalent to $E \subseteq \mathcal{M}_{\text{Assertion}}(A)(p)(p')$, and determines if the request requires less permissions than is granted by p . Figure 2 shows an example of the result of a least fixpoint computation.

5. Conclusion

In this paper we discussed an approach to modeling trust and authority based on role-based authorization and trust management for open and distributed environment. We first discussed our modeling effort, primarily focusing on the extension of role-based authority delegation from local domains to trusted domains. Then we presented a formal specification to instantiate our modeling approach as a proof-of-concept. Our future direction will be put on the issue of the specification of constraints in our current approach. This could be done by adding a component capable of handling the nonmonotonic characteristic, since constraints are considered to be as such.

References

- [1] M. Abadi, M. Burrows, and B. Lampson. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, 1993.
- [2] G.-J. Ahn and R. Sandhu. Role-based authorization constraints specification. *ACM Transactions on Information and System Security*, 3(4), November 2000.
- [3] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote trust-management system version 2. RFC 2704, September 1999.
- [4] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, CA, May 1996.
- [5] Y. H. Chu, J. Feigenbaum, B. LaMacchia, P. Resnick, and M. Strauss. REFEREE: Trust management for web applications. *Computer Networks and ISDN systems*, 29(8-13), September 1997.

- [6] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody. Using trust and risk in role-based access control policies. In *Proceedings of 9th ACM Symposium on Access Control Models and Technologies*, Yorktown, NY, June 2004.
- [7] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. RFC 2693, September 1999.
- [8] S. Farrell and R. Housley. An internet attribute certificate profile for authorization. Technical report, PKIX Working Group, June 2001.
- [9] C. Goh and A. Baldwin. Towards a more complete model for role. In *Proceedings of 3rd ACM Workshop on Role-Based Access Control*, Fairfax, VA, October 22-23 1998.
- [10] A. Herzberg, Y. Mass, J. Michaeli, D. Naor, and Y. Ravid. Access control meets public key infrastructure, or: assigning roles to strangers. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, May 2000.
- [11] ITU. *ITU-T Recommendation X.509. Information Technology: Open Systems Interconnection - The Directory: Public-Key And Attribute Certificate Frameworks*, 2000. ISO/IEC 9594-8.
- [12] T. Jaeger. On the increasing importance of constraints. In *Proceedings of 4th ACM Workshop on Role-Based Access Control*, pages 33-42, Fairfax, VA, October 28-29 1999. ACM.
- [13] N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed credential chain discovery in trust management. *The Journal Of Computer Security*, 11(1), February 2003.
- [14] J. Linn and M. Nystrom. Attribute certification: An enabling technology for delegation and role-based controls in distributed environments. In *Proceedings of 4th ACM Workshop on Role-Based Access Control*, Fairfax, VA, October 28-29 1999. ACM.
- [15] R. L. Rivest and B. Lampson. SDSI - a simple distributed security infrastructure. Technical report, September 1996.
- [16] R. Sandhu. Role activation hierarchies. In *Proceedings of 3rd ACM Workshop on Role-Based Access Control*, pages 33-40, Fairfax, VA, October 22-23 1998. ACM.
- [17] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38-47, February 1996.
- [18] R. S. Sandhu, D. Ferraiolo, and D. Kuhn. The NIST model for role-based access control: Towards a unified standard. In *Proceedings of 5th ACM Workshop on Role-Based Access Control*, Berlin, Germany, July 26-27 2000.
- [19] D. Shin and G.-J. Ahn. Role-based trust and privilege management. *Computer Science, Systems and Engineering Journal*, 20(6), November 2005.
- [20] D. Shin, G.-J. Ahn, and S. Cho. Role-based EAM using x.509 attribute certificate. In *Proceedings of Sixteenth Annual IFIP WG 11.3 Working Conference on Data and Application Security*, Cambridge, UK, July 29-31 2002.
- [21] D. Shin, G.-J. Ahn, S. Cho, and S. Jin. On modeling system-centric information for role engineering. In *Proceedings of 8th ACM Symposium on Access Control Models and Technologies*, Como, Italy, June 2-3 2003.
- [22] M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari. Certificate-based access control for widely distributed resources. In *Proceedings of 8th USENIX Security Symposium*, Washington, D.C., August 23-26 1999.
- [23] S. Weeks. Understanding trust management systems. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.
- [24] L. Zhang, G.-J. Ahn, and B.-T. Chu. A rule-based framework for role-based delegation. *ACM Transactions on Information and System Security*, 6(3), August 2003.
- [25] X. Zhang, S. Oh, and R. Sandhu. Pbdm: A flexible delegation model in rbac. In *Proceedings of 8th ACM Symposium on Access Control Models and Technologies*, Como, Italy, June 2003.