Towards Trust-aware Access Management for Ad-hoc Collaborations

Jing Jin, Gail-Joon Ahn, Mohammed Shehab, and Hongxin Hu
The University of North Carolina at Charlotte
{jjin,gahn,mshehab,hxhu}@uncc.edu

***Abstract*—In an ad-hoc collaborative sharing environment, attribute-based access control provides a promising approach in defining authorization over shared resources based on users' properties/attributes rather than their identities. While the user's attributes are always asserted by different authorities in the form of credentials, these authorities may not be accepted by the resource owner with the same degree of trust. In this paper, we present a trust-aware role-based authorization framework, called RAMARS_TM, to address both the access control and the trust management issues in such environment. Central to our approach is the dynamic role assignment based on a user's attributes, and trust management, as a special constraint, is in place to make trust decisions on a user's attributes. Required components and functions are identified and specified in our trust and access management policies. An architecture of prototype system implementation is also discussed.**

I. Introduction

In an ad-hoc collaborative environment, users who belong to different organizations spontaneously establish or join collaboration relationships, dynamically contribute and share data resources [10]. There is no pre-established global consensus of trustworthiness among all participating parties in such environment. The resource owner, also called *originator*, has to expect a large number of strangers, i.e. users that have no pre-existing relationships, trying to collaborate and request to share information. In this situation, what is important is not "Who exactly is this requester?", but "Do I trust this requester to share my resource?" As the identity alone does not imply privileges, it has made the traditional identity-based access control approaches ineffective in this situation. Instead, the properties/attributes possessed by the requester (e.g., employment status, citizenship, group membership and qualifications) will be more relevant to characterizing users and determining whether or not they should be trusted to conduct sensitive interactions involved in the collaborative information sharing.

Usually, user attributes are asserted in the form of credentials (i.e. certificates) that are issued by certain attribute certifiers or made by other entities as recommendations [17], and the same attribute may be asserted by different certifiers. As there is no central trusted attribute authority can be assumed in dynamic collaborations, credentials issued by different certifiers may not be trusted by the originator to the same extend and thus fail to assert the attributes of the user with the desired degree of trust, resulting in the denial of access to the originator's resource. As an example, an originator may only trust a requester's US citizenship presented in his passport which is certified by US government instead of a driver's license issued by a local DMV office. In addition, transitivity of trust is a common property in distributed environment where delegation is in place to construct a chain of trust extending to an end user through multiple intermediators. It is a critical issue to address the validation and criteria for trust evaluation over these trust propagation chains in order to determine the trustworthiness of the user attributes.

A. A Motivating Scenario

To better illustrate both the access control and trust management issues we have mentioned, a hypothetical scenario in the context of collaboration within the regional disease surveillance is presented below. We will use this example through the paper to demonstrate our proposed approaches.

Suppose Regional Emergency Department (RED) receives a dramatic increase in the number of patients with similar symptoms. This could be a sign of a new disease breakout or an incident of bioterrorism attack. To quickly diagnose the disease and facilitate the communication of this breakout, RED needs to securely share the medical data with other collaborating organizations. Without a priori knowledge of the exact persons in the collaborating organizations who may need access to investigate the data, RED defines the authorization through a set of required attributes that the user must possess. For example, security clearance guidelines require that the user must be a US citizen, and he must be a member of Disease Control Group. Suppose RED needs special experts from ABC National Lab (ABC) to help analyze the data. The user then must be affiliated with ABC and work as a role of "Investigator".

Suppose a user X is trying to access the medical data. He presents both his passport and driver's license to prove his US citizenship. Local Public Health Department issues a credential for his Disease Control Group membership. Since ABC has outsourced its human resource services to another company called AdminiStaff, along with a credential demonstrating the outsourcing relationship, AdminiStaff issues a credential asserting X's affiliation with ABC and his "Investigator" role. Upon receiving the access request with X's credentials, RED has to first decide on whether or not to trust these attributes based on X's credentials. And only trusted attributes can be used for determining X's access.

B. Our Objectives

Traditional role-based access control (RBAC) systems [18] achieve effective privilege management in a local domain, however, they do not address unknown users and trust relevant aspects encountered in collaboration settings. Nevertheless, Trust Management (TM) systems (e.g., SDSI/SPKI [6], KeyNote [3], [4], RT [13]) have been developed advocating attribute-based approach and managing delegation of authority to achieve the control in a decentralized fashion, which certainly can be adopted as a means to determine the trustworthiness among unknown collaborative entities.

In this paper, we propose a RAMARS_TM framework[1] that extends traditional RBAC with more flexible features inspired by research results from attribute-based access control and TM systems to provide an effective trust-aware role-based access management for secure resource sharing in ad-hoc collaborations. In particular, RAMARS_TM represents permissions for controlled sharing actions in terms of roles. The role assignment adopts a more dynamic and flexible attribute-based scheme, where the user's role(s) are determined by the set of attributes entitled to the user. A special Trust Management (TM) constraint is introduced to address major components and functions that contribute to determine the trustworthiness of user-attributes entitlements. Accordingly, a set of trust assessment policies are specified to direct the whole trust evaluation process. A system architecture is also proposed to implement the prototype of RAMARS_TM framework.

The rest of the paper is organized as follows. In Section II, we describe our proposed RAMARM_TM framework. We first introduce formal definitions of the necessary components in the framework and then focus on the trust management constraint in terms of key concepts and necessary policy specifications. The attribute-based role assignment policy and the evaluation is also briefly discussed. In Section III, we explain how the proposed framework can be realized in XACML policies by using the example discussed earlier. The prototype system architecture and implementation plan are also discussed in the same section. Section IV reviews related works that try to merge trust management with RBAC. Finally, Section V concludes the paper with future research directions.

II. RAMARS_TM Framework

In this section, we first present RAMARS_TM as an authorization framework that extends RBAC with attribute-based role assignment and the Trust Management (TM) constraint. The components and functions related to the trust evaluation of user-attributes entitlement, as abstracted in the component of TM constraint, are discussed and formally defined in another trust management layer.

A. RAMARS_TM in the Authorization Layer

As shown in Figure 1, RAMARS_TM adopts basic components from RBAC [18] for privilege management and introduces new components to accommodate the new scheme of role assignment through the user-attributes entitlement. We follow the conventional approach to defining them using sets, relations, and functions.

Definition 1: The following is a list of original RBAC components:

- U,R,P and S are sets of users, roles, permissions, and sessions, respectively.
- $PA \subseteq P \times R$ is a many-to-many permission to role assignment relation.
- $RH \subseteq R \times R$ is a partial order on R, written as $\preceq$.

Definition 2: The following are additional components introduced in RAMARS_TM model:

- $ATTR$ is a set of attributes involved in the system. $ATTR = \{attr_1, \ldots, attr_n\}$.
- AS is a collection of attribute sets. $\{as_1, \ldots, as_s\}$ where $as_i = \{attr_i, \ldots, attr_t\} \subseteq ATTR$, $i \in [1, s]$.
- $ETL \subseteq U \times AS$ is a many-to-many user-attributes entitlement relation.
- $RA \subseteq ETL \times R$ is a many-to-many user-attributes entitlement to role assignment relation.
- usr_attrs_etl: $S \rightarrow ETL$ is a function mapping each session s_i to a single user-attributes entitlement $usr\_attrs\_etl(s_i)$.
- $roles$: $S \rightarrow 2^R$ is a function mapping each session s_i to a set of roles $roles(s_i) \subseteq \{r | (\exists r' \geq r)[usr\_attrs\_etl(s_i), r' \in RA]\}$.
- $role\_etls$: $R \rightarrow 2^{ETL}$ is a function mapping each role r_i to a set of user-attributes entitlements $role\_etls(r_i) \subseteq \{etl | (etl, r_i) \in RA\}$.

B. RAMARS_TM in Trust Management Layer

Trust Management constraint is the most important component to convey issues in determining the trustworthiness of user-attributes entitlements. We would like to elaborate more details of this constraint by first introducing a few related key concepts and functions.

Definition 3: (**Entities**). The set of entities $E = \{e_1, \ldots, e_n\}$ are defined to generalize all related parties (individuals and/or organizations) in the collaborative environment. As a subset, *individual users* $U \subseteq E$, is a special subset of entities, who need to be authorized to access to the resource. In our example, the requester *X* is an individual user to whom the access permission should be granted.

Definition 4: (**Attributes**). The set of attributes are defined as $ATTR = \{attr_1, \ldots, attr_n\}$, where each $attr_i$ ($i \in [1, n]$) is represented in the form of $(Aname, Value)$ pair, for instance, `(citizenship, US)`.

An individual user claims the entitlement of an attribute by presenting supportive **credential**(s). We consider two types of credentials in our framework, namely *attribute credential* and *delegation credential*. An *attribute credential* is a statement issued by a certifier to entitle certain attribute(s) to a user. In our example, both the passport and the driver's license are attribute credentials to assert the `X-(Citizenship, US)`

[1]RAMARS_TM stands for Role-based Access Management for Ad-hoc Resource Sharing with Trust Management. This is a continuous work of our previous RAMARS framework [10].

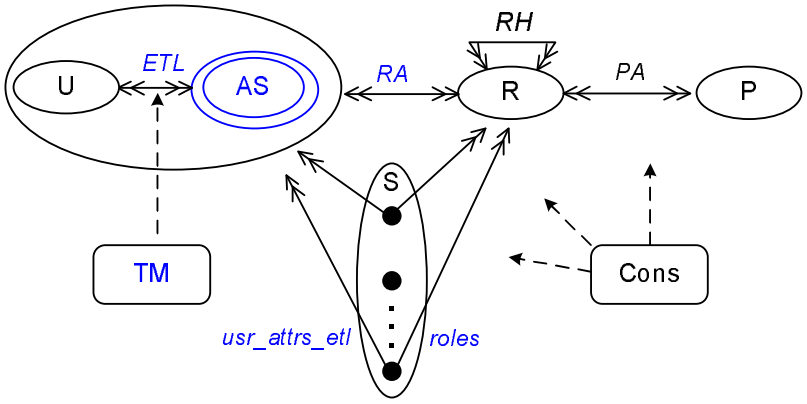

Fig. 1. RAMARS_RM Authorization Layer

entitlement. A *delegation credential* is a statement specifying the delegation relationship between two entities to transfer assertion rights over certain attribute(s), one as the delegator (certifier) and the other as the delegatee (credential holder) regarding to certain attribute(s). By outsourcing, `ABC` delegates attributes of `(affiliation,ABC)` and `(role, Investigator)` to `AdminiStaff`. In other words, `ABC` trusts and authorizes `AdminisStaff` to issue attribute credentials. Besides, the certifier may need to specify certain constraints over their issued credentials. We define the *context* as an additional component in the credential specification to include all circumstance constraints that may be used to determine the validity of a credential. As addressed in [1], the validity period is a basic context constraint that can be defined as $Ctx_I = \{[start, end] | start, end \in Time\_Instances \cup \infty, start \leq end\}$. Another important context constraint is to define how further a credential can be delegated. A non-negative integer is defined as the maximum delegation depth for this context constraint. Formally, $Ctx_D = \{n | n \in Z, n \geq 0\}$. By combining these two context constraints, we can define a new context $Ctx_{I \times D} \subseteq Ctx_I \times Ctx_D$. New context constraints can be defined in similar ways when new validation requirements appear. To summarize, we define a credential as follows:

Definition 5: (**Credential**). A credential is defined as a tuple of $cred = (holder, attrs, certifier, Ctx)$, where $holder, certifier \in E$, $attrs \subseteq ATTR$, and $Ctx \subseteq Ctx_1 \times Ctx_2 \times \ldots \times Ctx_n$. We use the dot notation to refer to its elements, such as $cred.holder$. And we define a function of $max\_depth(cred) = cred.Ctx.Ctx_D$ to denote the delegation depth constraint.

The sequence of delegation credentials and an attribute credential asserting the same attribute(s) construct a chain of trust propagation extending to the end user. We define this chain as an assertion path.

Definition 6: (**Assertion path**). An assertion path $ap[attrs]$ regarding the $attrs$ is a sequence of credentials $cred_1 \rightarrow cred_2 \rightarrow \cdots \rightarrow cred_i \rightarrow \cdots \rightarrow cred_n$, where $n \geq 1$, $cred_i = (holder_i, attrs, certifier_i, Ctx_i)$, $cred_i.holder = cred_{i+1}.certifier$ for all $i \in [1, n-1]$. Simply, $ap[attrs] = cred_1 cred_2 \ldots cred_n$. And we define a function to retrieve the number of credentials in an assertion path, i.e. $depth(ap[attrs]) = n$.

- If $depth(ap[attrs]) = 1$, then the $attrs$ is directly asserted by an attribute certificate.
- Otherwise, the $attrs$ is asserted indirectly through a delegation chain including an ordered list of credentials in $ap[attrs]$.

A user's attributes may be directly and/or indirectly asserted through different assertion paths with different degree of trust. When inspecting the trust on assertion paths, not all possible paths need to be considered. We define the following two validation functions to evaluate the validity of the assertion paths, so that invalid paths can be discarded before the trust evaluation.

Definition 7: (**Credential validation function**). The validation function is defined as a boolean function of:

$validate(cred, EN) \overset{cred.Ctx}{\rightarrow} \{true, false\}$, where $cred$ is a credential to be evaluated, and EN is the current environmental parameters captured at runtime according to the validation constraints defined in $cred.Ctx$.

As an example of evaluation using the instant of validity period, suppose $cred.Ctx.Ctx_I = [start, end]$, $validate(cred, EN) = true$ iff $start \leq currentDate(EN) \leq end$.

Definition 8: (**Assertion path validation function**). Let $ap[attrs] := cred_1 cred_2 \ldots cred_n$ be an assertion path. The validation function is defined by a boolean function:

$validate(ap[attrs], EN) \rightarrow \{true, false\}$.

The function evaluates the validity using the algorithm shown in Figure 2, where each $cred_i$ ($i \in [1, n]$) is first validated using the credential validation function, then the whole assertion path is validated against the delegation constraints.

Definition 9: (**Trust level**). Trust level is a measure indicating the degree of trust that an originator may put on an assertion path regarding the asserted attributes. The set of trust levels is defined as a partial order $(TL, \preceq)$.

In our example, assume both the requester X's passport and driver's license are valid to assert his `(citizenship,`

```
Algorithm validate
  Input: ap[attrs], EN      /* ap[attrs] is a particular assertion path to be validated, EN is the
                               environmental parameters */
  Output: true if valid, otherwise false
/* check validity for direct assertion */
IF depth(ap[attrs]) = 1 THEN    /* if there's only one attribute credential in the assertion path */
    result = validate(cred_i, EN);
    return result;              /* validate the credential and return the result */
/ * check validity for indirect assertion */
ELSE
    FOR each (credi∈ap[attrs]) DO
        result = validate(cred_i, EN);
        IF result = false THEN
            return false;       /* validate each individual credential */
        ELSE IF max_depth(cred_i)< (depth(ap[attrs])-i) THEN
            return false;       /* validate the whole path against delegation constraints */
return true;
```

Fig. 2. Algorithm for Assertion Path Validation Function

US) attribute. However, RED, as the originator, may trust the passport more than the driver's license. Similarly, the depth of an assertion path may also affect the trust level for indirect assertions.

Trust level serves as an effective mechanism for an originator to subjectively rank and compare different assertion paths regarding the asserted user-attributes entitlement. Yet the ultimate question in RAMARS_TM is whether the user-attributes entitlement can be trusted by the originator for further authorization. The originator has to make the final decision on the trustworthiness of the entitlements given the references of trust levels achieved by different assertion paths. In our example, the X-(citizenship,US) entitlement asserted by the driver's license (DMV) may not meet the required trust level, and thus can not be trusted.

Definition 10: (**Trust level assessment function**). We define the function mapping the trust level of each assertion path as:

$$trustAssessment : AP[Attrs] \overset{TAP.TL}{\rightarrow} TL.$$

Definition 11: (**Trustworthiness assessment function**). We define the function of mapping the trust level of each assertion path to a boolean trust decision:

$$trustDecision : AP[Attrs] \times TL \overset{TAP.TD}{\rightarrow} \{true, false\}.$$

Trust level and trustworthiness assessment functions rely on a policy component to be available for the evaluation. We define the trust assessment policy (*TAP*) as a set of rules that an originator defines to govern the process of assigning trust levels to assertion paths (*TAP.TL*) and to make the trust decision on the entitlements based on a predefined threshold of the minimum required trust level (*TAP.TD*). The details of *TAP* specification will be discussed in the subsequent section. Different policies will result in different trust decisions even for the same set of supportive credentials of the user-attributes entitlement. The trust assessment policy is subjective and discretionary to the originator. In addition, as the collaboration relationships may change over time, an originator needs to adjust its trust assessment policy accordingly to accommodate changes in the environment and new trust requirements that may emerge.

C. Trust Management Policies and Evaluation Algorithm

The attribute credential and delegation credential are used to express the assertions of user attributes and the delegation relationship between two certifiers over certain attributes, respectively. The validation constraints (as represented in *cred.Ctx*) make the credentials policy neutral, and thus require clear syntax and semantics on the policy specification. In addition, *TAP*, as mentioned earlier, is a necessary policy component to define the rules for trust assessment. Therefore, our trust management policy framework includes the attribute credential policy (*ATTR Policy*), delegation policy (*DLGT Policy*), and trust assessment policy (*TAP*).

Attribute credential policy (*ATTR Policy*) specifies the *context* constraints under which a *certifier* authorizes/asserts certain *attributes* to a *user*. The general definition of *ATTR Policy* is a tuple

$$\langle holder, attrs, certifier, Ctx_D | Ctx_I | \ldots \rangle.$$

It shares similar elements to the credential specification (Definition 5), but emphasizes on the *Ctx* component, where context constraints are specified as different types of constraint rules, such as time interval rules, delegation rules, etc.

Delegation credential policy (*DLGT Policy*) specifies the constraints under which a certifier as a delegator delegates the right over certain *attributes* to another certifier as a delegatee. A *DLGT Policy* is defined as the same format of an *ATTR Policy*.

Trust assessment policy (*TAP*) is used in two types of functionalities: to determine the trust level of an assertion path regarding certain attribute(s); and to make the trust decision on the user-attributes entitlement. We realize the two functionalities using two policies referred to as *TAP.TL* and *TAP.TD*, respectively.

We consider three major factors for an originator to determine the trust level of an assertion path regarding certain attribute(s): (1) the certifier of the attribute credential, with which the originator may have different trust relationships; (2) the depth of the delegation chain for the asserted attribute if delegation credentials are presented; (3) the number of certifiers asserting the same attribute if recommendations are considered. *TAP.TL* is defined as: $\langle ap[attrs], certifier | AP\_depth | No\_certifiers, tl \rangle$. The semantics of *TAP.TL* is that a trust level *tl* is assigned to an assertion path $ap[attrs]$ based on conditions of the *certifier*, the *delegation_depth*, and/or the *number of certifiers*.

TAP.TD policy specifies the rule that the minimum required trust level an asserted attributes has to achieve in order to be trusted by an originator. *TAP.TD* is specified as a tuple of $\langle ap[attrs], req\_tl \rangle$, where req_tl is the minimum required trust level for which the asserted user-attributes entitlement to be trusted.

Given a set of supportive credentials, we design an algorithm, called evalTrust, to evaluate the trustworthiness of the claimed entitlements (Figure 3). The algorithm works as follows. In Step 1, the supportive credentials are categorized

```
Algorithm evalTrust
  Input: attrs, CredSet, EN, TAP     /* attrs is the attributes to be evaluated, CredSet is the supportive credential set, EN
                                        is current environmental settings, TAP is Trust Assessment Policy */
  Output: true if attrs is trusted, false otherwise

/* Step 1: Finding credential paths and path validation */
Paths := findAssertionPaths(attrs, CredSet);
FOR each (p∈Paths) DO
    IF validate(p, EN) ≠ true THEN
        Paths.remove(p);                    /* validate each path p in Paths, and remove invalid ones */

/* Step 2: Trust level assessment */
Set TLs := null;
FOR each (p∈Paths) DO
    tl = trustAssessment(p, TAP.TL);        /* assign trust level tl to each valid p in Paths according to TAP.TL policy */
    TLs.add(tl);                            /* add tl to TLs */

/* Step 3: Trustworthiness evaluation */
FOR each (tl∈TLs) DO
    IF makeDecision(attrs, tl, TAP.TD) = true THEN
        return true;                        /* return true if any tl achieves true in trust decision against TAP.TD policy */
    return false;                           /* return false otherwise */
```

```
Algorithm findAssertionPaths
  Input: CredSet, attrs          /* CredSet is the available credential set, attrs is the asserted attributes */
  Output: APs                    /* a set of assertion paths APs that can be derived from CredSet for the given attr */

Set relevantCreds := null;
FOR each (c∈CredSet) DO
    IF c.Attrs = attrs THEN
        relevantCreds.add(c);             /* add all relevant credentials asserting attrs to relevantCreds set */

FOR each (c∈relevantCreds) DO
    IF c.Ctx.Ctx_D = 0 THEN
        List ap.add(c);                   /* initiate a new assertion path for each single attribute credential */
        APs.add(ap);                      /* add the path to APs */
        remove(c, relevantCreds);

WHILE length(relevantCreds > 0) DO
    FOR each (ap∈APs) DO
        c := ap.get(length(ap)-1);        /* retrieve the last credential in the assertion path */
        FOR each (cred∈relevantCreds) DO
            IF cred.Holder = c.Certifier THEN
                ap.add(cred);             /* find and add the immediate precedent delegation credential to the path */
                remove(cred, relevantCreds);   /* remove the just added delegation credential from relevantCreds */

FOR each (ap∈APs) DO
    reverseElements(ap);                  /* reverse all elements in each ap to get the correct order of an assertion path */

return APs;
```

Fig. 3. Trustworthiness Evaluation Algorithm

into a set of assertion paths ($Paths$) regarding the asserted attributes through another algorithm `findAssertionPaths`. Each path p is validated using the function $validate(p, EN)$ against *ATTR* policies and/or *DLGT* policies as defined in Definition 8. Invalid assertion paths are discarded. In Step 2, each of the valid assertion paths is evaluated against *TAP.TL*, resulting in a trust level (tl) being assigned. This procedure utilizes the trust level assessment function defined in Definition 10. Finally in Step 3, these trust levels achieved by different assertion paths are evaluated against the trust decision policy (*TAP.TD*) to make the final decision on whether the claimed user-attributes entitlements are trusted for further role assignment or not. A value of *true* is returned if any of the assertion paths achieves satisfiable trust level, so that the claimed user-attributes entitlements asserted by this assertion path will be trusted and used for further role assignment. This step utilizes the trustworthiness assessment function defined in Definition 11.

D. Role Assignment Policy and Evaluation

In our RAMARS_TM framework, the trust management constraint is utilized to derive the trusted user-attributes entitlements, and the role assignment is only based on these trusted user-attributes entitlements. The simplest form of role assignment policy (***RA Policy***) is specified as $\langle r, etls \rangle$, to define the required user-attributes entitlements for a user to

be assigned to a particular role. This is generalized using the following function:

Definition 12: (**Role assignment function**). We define the function as:

$roleAssignment(etl) \stackrel{RA}{\rightarrow} 2^R$, where etl is the set of trusted entitlements, RA is the *RA Policy*, and the function returns a set of roles assigned to the user-attributes entitlement.

Constraints that are addressed in RBAC models such as Separation of Duty (SoD), prerequisite and cardinality [18], and later recognized temporal and location based constraints [2], [12] can also be specified in the *RA Policy* to meet various access management requirements involved in the ad-hoc collaboration environment. The research on these constraints are out of the scope of this paper.

III. Realizing the RAMARS_TM Framework and Implementation Plan

XACML [15] is an OASIS standard that describes a general policy language used to protect resources as well as an access decision language. Recently published XACML3.0 [16] has included delegation concepts. The draft defines two types of policies, *access policies* and *administrative policies*, to differentiate normal authorization policies from delegation policies. The delegation chain is derived and validated through a process of policy reduction back to the original delegator's policy of the delegation chain. Using these features, we design a set of XACML-based policies to implement our RAMARS_TM policy framework. Figure 4 illustrates the policy examples to realize the motivation scenario we discussed earlier.

The *ATTR Policy* specifies a passport attribute credential/policy that the `US government`, as the policy certifier, asserts `X-(citizenship, US)` entitlement, with the interval context constraint between `12/31/2002` and `12/31/2007`. The *DLGT Policy* specifies that `ABC` delegates to `AdminiStaff` the right of `(affiliation,ABC)` and `(role,Investigator)` attributes. Within the delegation, `ABC` also specifies the $max\_dlgt\_depth = 1$ as the delegation constraint to restrict further delegations by `AdminiStaff`. The *TAP.TL* policy realizes the trust level assessment policy defined by the originator (`RED`). Inside the policy, Rule 1 specifies that `RED` trusts `US Government` with a `high` level to assert `(citizenship, US)`; and Rule 2 specifies that `RED` trusts `ABC` with the maximum delegation depth ≤ 2 with a `medium` level of trust. With this policy being defined, the entitlement of `X-(citizenship,US)` asserted by `US Government` will be assigned to a `high` level of trust. And a `medium` level of trust will be assigned to *X*'s affiliation and role attributes. In *TAP.TD* policy, Rule 1 specifies that the `(citizenship, US)` attribute will be trusted when its achieved trust level is `high`, where Rule 2 specifies that the `affiliation` and `role` attributes will be trusted when its achieved trust level is equal or higher than `medium`. With this policy being defined, *X*'s US citizenship, affiliation and role attributes are all trusted by `RED` and can be promoted for further role assignment evaluation. Finally, the *RA policy* specifies that a user is assigned to a `Collaborator` role with all the following attributes: `(citizenship, US)`, `(affiliation, ABC)`, `(role, Investigator)` and `(membership, DCG)`.

As part of our on-going prototype implementation effort, JDK1.5 core packages as well as other necessary Java libraries are used to develop the RAMARS_TM Engine in accommodating the evaluation procedures illustrated in the `evalTrust` algorithm (Figure 3). We adopt SICS's XACML3.0 PDP implementation [21] to accommodate the policy evaluation functionalities involved in the procedures of credential validation, trust level assignment and trust decision making. As we have identified that credential are policy neutral when the context constraints are in place to determine the validity of the credential itself. We directly use XACML-based *ATTR Policy* and *DLGT Policy* to represent the attribute and delegation credentials, respectively, so that our framework can be easily and consistently implemented utilizing the policy evaluation mechanism provided by XACML. These policies are also flexible to be encoded into other implementation mechanisms, such as X.509 attribute certificates [8], or conveyed through SAML assertions [7], [14]. Upon receiving the trust and authorization decisions from RAMARS_TM Engine, the implementation of PEP is subject to the specific data sharing application. As long as the communication interface and protocol are well defined [11], the RAMARS_TM Engine is able to serve as the general trust management and authorization service for any data sharing applications.

IV. Related Works

There are a number of approaches being published in recent years to merge TM and RBAC. [13] and [20] consider TM as a form of distributed access control managing role-related credentials exchange and trust propagation across domains through distributed policy statements. However, these approaches are confined by assuming the "role" as the major mediator that can be delegated and associated in relevant domains. Our approach, driven by the general access management requirements in collaborative environments, focuses on a more general scale of trust management issues, where the trust decision making relies on not only the role-related credentials, but more general-purpose credentials, such as affiliations and qualifications.

In [5], the authors attempt to extend conventional RBAC by introducing the notion of trust. In their scheme, users are first mapped to different trust levels, which are then indirectly mapped to roles in RBAC. As the trust level is closely associated with the role assignment, extra burden and complexity have been laid on the design of role and role hierarchy construct, where trust requirements have to be taken into consideration. It interferes with the extendibility of both RBAC and trust management, and the discrepancies between the dominance in trust levels and role hierarchies are difficult to be reconciled.

The closest work to our approach is [19], where authorizations are determined based on user's attributes. Trust evaluation is used to determine the degree of trust in the

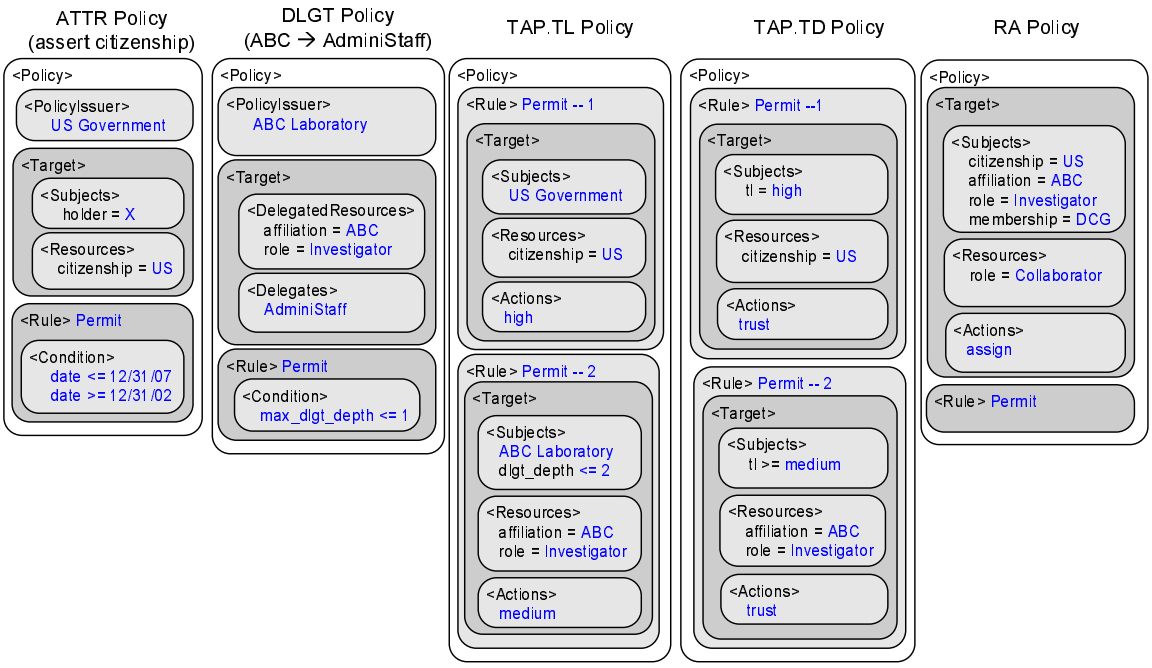

Fig. 4. XACML Policy Examples

attributes claimed by the user based on the credentials supplied by the user. However, the direct association of role definition with the trust evaluation constraint poses similar limitations as in [5]. And the work lacks details in providing concrete criteria and solutions for trust relationship management and trust evaluation for user's credentials.

V. Conclusion and Future Work

In this paper, we have presented the RAMARS_TM framework in order to support secure resource sharing in ad-hoc collaborations. In our framework, the authorizations of unknown collaborators are determined based on their trusted user-attributes entitlements. We identified the necessary elements and functions in the process of trust evaluation, and a set of trust management policies are specified accordingly. The system architecture outlined a mechanism to develop an integrated trust and access management prototype realizing the proposed RAMARS_TM framework.

As our future work, we would like to fully implement the RAMARS_TM Engine into our secure resource sharing tool, ShareEnabler [9], [10], to provide trust-aware access management for both P2P and Grid collaborative sharing environments. In addition, trust negotiation [22], [23] will be explored as an iterative process of disclosing access control policies and exchanging credentials between the originator and the requesting party. In particular, the requesting party should be able to gradually discover the originator's authorization policies, in order to present the right credentials that guarantee the access.

Acknowledgment

The work was partially supported by the grants from National Science Foundation (NSF-IIS- 0242393) and Department of Energy Early Career Principal Investigator Award (DE-FG02-03ER25565).

References

[1] I. Agudo, J. Lopez, and J. A. Montenegro. Attribute delegation based on ontologies and context information. In *Prceedings of 10th IFIP TC-6 and TC-11 Conference on Communications and Multimedia Security (CMS'06), LNCS 4237*, pages 54–66. Springer, 2006.

[2] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca. GEO-RBAC: a spatially aware RBAC. In *Proceedings of 10th Symposium on Access Control Models and Technologies (SACMAT)*, pages 29–37, 2005.

[3] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote trustmanagement. RFC2704, 1999.

[4] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[5] S. Chakraborty and I. Ray. TrustBAC: integrating trust relationships into the RBAC model for access control in open systems. In *Proceedings of the 11th ACM symposium on Access control models and technologies (SACMAT)*, pages 49–58, 2006.

[6] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI certificate theory. RFC 2693, 1999.

[7] Internet2. Opensaml 1.1 – an open source security assertion markup language implementation. http://www.opensaml.org/.

[8] ITU. ITU-T Recommendation X.509. Information technology: Open system interconnection – the directory: Public-key and attribute certificate framework. ISO/IEC 9594-8, 2000.

[9] J. Jin and G.-J. Ahn. Policy-driven access management for ad-hoc collaborative sharing. In *Proceedings of 2nd International Workshop on Pervasive Information Management (PIM)*, 2006.

[10] J. Jin and G.-J. Ahn. Role-based access management for ad-hoc collaborative sharing. In *Proceedings of 11th Symposium on Access Control Models and Technologies (SACMAT)*, 2006.

[11] J. Jin and G.-J. Ahn. Towards secure information sharing and management in grid environments. In *Proceedings of 2nd IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2006.

[12] J. B. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23, 2005.

[13] N. Li and J. C. Mitchell. RT: A role-based trust-management framework. In *Proceedings of The Third DARPA Information Survivability Conference and Exposition (DISCEX III)*, pages 201–212, 2003.

[14] OASIS. SAML 2.0 profile of XACML. http://docs.oasis-open.org/xacml/access_control-xacml-2.0-saml_profile-spec-cd-02.pdf, November 2004.

[15] OASIS. XACML 2.0 core: extensible access control markup language (XACML) version 2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, February 2005.

[16] OASIS. XACML 3.0 administrative policy working draft 10, December 2005.

[17] A.-A. Rahman. The PGP trust model. *The Journal of Electronic Commerce*, 1997.

[18] 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.

[19] B. Shafiq, E. Bertino, and A. Ghafoor. Access control management in a distributed environment supporting dynamic collaboration. In *Proceedings of the 2005 workshop on Digital identity management (DIM'05)*, pages 104–112, November 2005.

[20] D. Shin and G.-J. Ahn. Role-based privilege and trust management. *Computer Science, Systems & Engineering Journal*, 20(6), November 2005.

[21] Swedish Institute of Computer Science. XACML 3.0 administrative policy support (beta version). http://www.sics.se/spot/xacml_3_0.html, 2006.

[22] W. Winsborough and N. Li. Towards practicl automated trust negotiation. In *Poceedings of IEEE Workshop on Policies for Distributed Systems and Networks*, pages 92– 103, 2002.

[23] T. Yu and M. Winslett. A unified scheme for resource protection in automatic trust engotiation. In *proceddings of the IEEE Symposium on Securiy and Privacy*, May 2003.