Journal of Computer Security 20 (2012) 223–257
DOI 10.3233/JCS-2012-0446
IOS Press

Policy-driven role-based access management for ad-hoc collaboration

Gail-Joon Ahn [a,*], Jing Jin [b] and Mohamed Shehab [c]

[a] *Arizona State University, Tempe, AZ, USA*
E-mail: gahn@asu.edu
[b] *Deutsche Bank Global Technologies, Frankfurt, Germany*
E-mail: jjin@uncc.edu
[c] *University of North Carolina at Charlotte, Charlotte, NC, USA*
E-mail: mshehab@uncc.edu

Ad-hoc collaboration is a newly emerged environment enabling distributed collaborators to share resources. The dynamic nature and unique sharing pattern in ad-hoc collaboration poses great challenges for security services to accommodate both access control and trust management requirements in providing controlled resource sharing. In this paper, we propose a comprehensive, integrated and implemented access management framework, called RAMARS, for secure digital information sharing in ad-hoc collaboration. Our framework incorporates a role-based approach to leverage the originator control, delegation and dissemination control. A trust awareness feature is integrated for dynamic user-role assignment based on user attributes. The access control polices are formally specified, and a peer-to-peer scientific information sharing system – ShareEnabler – is presented to demonstrate the feasibility of our approach. The performance evaluation of our prototype system with potential system improvements is also discussed.

Keywords: Ad-hoc collaboration, access management, security architecture, XACML, policy enforcement

1. Introduction

The rise of Internet and Web technologies has enabled traditional scientific collaborations to turn outward and connect distributed participants across enterprises and research institutes. By removing the geographical distance barriers, scientists and engineers from different organizations are able to establish collaboration relationships and share information correspondingly. Under many circumstances, the establishment of collaboration relationship is highly dynamic and may vary tremendously in purpose, scope, size, duration and the number of involved participants. We refer this type of collaboration as ad-hoc collaboration [24–26]. Ad-hoc collaboration allows individual participants who belong to many different organizations to spontaneously establish or join collaborations, and dynamically perform a variety of

*Corresponding author: Gail-Joon Ahn, Arizona State University, 699 S. Mill Ave., Tempe, AZ, USA. E-mail: gahn@asu.edu.

activities such as communication, information sharing, cooperation, problem solving and negotiation [3,44]. Compared to well-structured collaborations, the formulation of ad-hoc collaboration is essentially more transient. Interactions among collaborating users are always unpredictable, and there is no pre-established global consensus of trustworthiness among all participating parties. As a result, it requires a more light-weighted infrastructure without pre-configured environments or central management authorities in ad-hoc collaboration. Among the various activities performed in ad-hoc collaboration, in this paper we focus on the digital information sharing.

Given all the diverse contexts and supporting infrastructures of ad-hoc collaboration, achieving effective access control is a critical requirement. The sharing of sensitive information is necessarily to be highly controlled by defining what is shared, who and under which condition is allowed to share. In multidomain collaborative environments, users without pre-existing relationships may try to collaborate and request the information. It is required for a data provider to be able to cope with a large number of strangers and guarantee the information be released only to trusted collaborators within the community. And the re-dissemination of such information needs to be strictly regulated as well. As the traditional identity-based access control approaches (e.g., MAC, DAC and RBAC) are considered ineffective for authorizing strangers in distributed environments, many delegation and trust management approaches have been proposed using credentials to delegate permissions and propagate administrative authorities [13,29,42,55]. The process of making access control decisions involves finding a chain of credentials that delegates the authority from the source to the requester. However, the decision making in these approaches neglects certain important properties of a delegation chain, which would dramatically affect the trustworthiness of the requester. For example, the length of a credential chain indicates the extent to which the trust is propagated from the trusting entity to the requester. The longer a credential chain is for the delegated authority, the weaker trustworthiness should be placed on the requester. In addition, in ad-hoc collaboration the partnerships between the authorization entity and collaborating parties vary dramatically depending on different collaboration purposes. A close and stable relationship deserves a higher level of trust over the authority delegation, while a newly established collaboration may achieve less trust as intended. Therefore, the delegation parties that formulate the credential chain also contribute to the final trust decision as well as the authorization over the requester. The above observations indicate that the binary trust paradigm implemented in general trust management approaches are inadequate for access control in distributed collaborative environments. Especially, there is a need to design a comprehensive access management framework that is general and flexible enough to cope with the special access control and trust management requirements associated with ad-hoc collaboration. In this research, we would make one step towards this direction. In particular, we advocate a policy-driven approach to enabling the originator control for the access and dissemination of the sensitive information in ad-hoc collaboration. Our framework coherently incorporates an extended role-based approach with delegation and trust awareness features for an originator to

dynamically define the scope of information dissemination and delegate information sharing capabilities to unknown users based on multi-level trust relationships. We formally define the policy components and illustrate the prototype implementation we have developed.

This article extends the work in [24–26], where the access control model and prototype system have been proposed. Especially, we extend the trust awareness feature in the framework by articulating the trust inference problem and we propose an algorithm to determine the trustworthiness of remote users beyond the compliance checking. Also, we further develop our policy framework to support trust-aware access management. Moreover, we enhance the prototype system with support of trust evaluation and dissemination control. In addition, we elaborate our performance evaluation including workload generation and overhead analysis. Through our evaluation, we also demonstrate the scalability and robustness of our system.

The rest of the paper is organized as follows. In Section 2, we overview generic access control requirements in ad-hoc collaboration through a motivating example. Our RAMARS access management framework is introduced in Section 3. The policy specification and evaluation is discussed in Section 4. The ShareEnabler prototype system is introduced in Section 5 with the performance evaluation in Section 6. We review other related work that dealt with authorization and trust issues in collaborative environments in Section 7. Section 8 concludes the paper with future research directions.

2. Problem domain analysis

We first introduce the characteristics of ad-hoc collaboration through a hypothetical scenario in context of public health surveillance.

The public health surveillance requires collaboration and information sharing among health care providers, laboratories and government agencies for real-time outbreak detection and monitoring. Abnormality discovered at any sector would trigger the immediate establishment of collaboration. Suppose Regional Medical Center (RMC) receives a dramatic increase in patient encounters with suspicious flu-like symptoms. This could be a sign of new disease breakout. The medical data must be shared immediately with collaborating laboratories to identify the causing virus. Professionals in other hospitals and clinics may also need to obtain the information for more efficient diagnose and treatments. In addition, regional public health authorities have to be notified to monitor the epidemics. However, sharing of sensitive medical data is restricted by privacy protection regulations and federal laws. RMC is responsible to apply appropriate conditions to strictly control the data access and dissemination. However, as users from multiple external agencies may access the data, RMC faces great challenges to decide the trustworthiness and authorize these unknown users.

From the scenario above, we refer the data resource owner (RMC) as an *originator* and the recipients of the data resource as *collaborators*. An originator, individually or organizationally, provides the information to be shared with other collaborators within the community. We further differentiate the concepts of *collaborating organizations* and *collaborating users*. In this paper, we assume an originator has limited trust on the collaborating organizations based on pre-established relationships. The collaborating organizations do not directly share the data. Instead, they carry out responsibilities to identify and nominate users from their perspective domains, and the individual users within these organizations are the actual recipients of the data. We call these users as *collaborating users*. In our scenario, the laboratories, clinics and government bureaus are collaborating organizations, while the users within these organizations (e.g., lab technicians, doctors and government staffs) are collaborating users who need to be authorized to access and/or disseminate the data to fulfill their duties.

- *Originator control.* As shown in the scenario, the collaboration is always triggered "on the fly" by sudden events. This feature results in frequent changes in the scope and structure of collaboration. Users may leave and new users may join the collaboration at any time. The space of collaboration spans across organizational boundaries and cannot be determined by conventional attributes. Compared to the ever-changing collaboration relationships and participants, the data being disseminated and its origination are relatively static. Entities towards the data resource can be separated into two parties: the resource owner (or *originator*) who provides the data, and the resource recipients (or *collaborating users*) who consume the data. A semicentralized control strategy should be applied where an originator has ultimate authorities over her data resource. In particular, an originator is responsible to define who is authorized to access the data and to what extent the data can be distributed. We refer this as an originator's *collaborative sharing control domain*.
- *Flexible and manageable access control.* Since ad-hoc collaboration may involve a large number collaborating users which are unknown to the originator, the authorizations cannot be explicitly specified based on users' identities. Instead, appropriate abstractions must be applied to effectively and efficiently manage these unknown users and authorize their privileges. More likely, users should be authorized in an indirect fashion based on their attributes or properties, such as security clearance, affiliation, membership and qualifications.
- *Trust management and delegation.* As there is no central administrative point or global agreement of trust in the community, an originator is responsible to determine the trustworthiness of delegated collaborating parties. Especially, a user attribute may be asserted by various forms of credentials, and the attribute authorities who issue these credentials may not be trusted by an originator to the same extent. For instance, a user's US citizenship can be asserted by a passport, by his birth certificate or by his driver's license. The originator may only

trust the passport as the legitimate credential for the attribute in some cases. Therefore, the criteria for delegating and determining the trustworthiness of user attributes should also be clearly specified as part of the access control requirements.

- *Effective dissemination control.* Digital information sharing involves data transmission among participating parties. Ideally, an originator should be able to specify access management policies, and let the authorization infrastructure enforce these policies on the originator's behalf along with the information dissemination. Therefore a policy-driven approach with distributed policy propagation and enforcement scheme is required in the sharing infrastructure.

There are numerous applications and infrastructures available to support collaborative information sharing, such as P2P networking and Grid computing. Yet in order to identify the generic access and dissemination control requirements in the environment, we must analyze the inherent information sharing patterns despite of the sharing infrastructures. For any type of information sharing process, the originator (*OR*) and the collaborating user (*Col*) are two major actors, serving as the information provider and recipient, respectively. In general, collaborative information sharing always involves a set of generic activities between these two actors including *resource publication*, *resource discovery*, *resource access* and *resource dissemination/redissemination*. An originator first publishes the data resource in the collaborative community with certain metadata information to describe the resource. A collaborating user could discover the resource through queries based on a description of desired properties. The user can further request to share the resource by either directly accessing or indirectly obtaining a copy of the data resource. And the data resource may be further re-disseminated thereafter. In terms of information dissemination, we consider the originator as an *initial disseminator* since she triggers the initial resource distribution. And we call a collaborating user who is authorized to further disseminate pre-obtained resource copies as a *designated disseminator*. Figure 1(a) illustrates the use case with critical procedures for resource dissemination between an originator and a collaborating user. We believe these procedures are generic operations that must be supported by any collaborative sharing infrastructures. And access control to each step of information sharing activity should be clearly defined and effectively enforced accordingly.

In terms of the overall information dissemination within the community, Fig. 1(b) shows an ideal pattern of highly regulated dissemination and re-dissemination for a particular data resource. We assume there is a virtual collaborative sharing domain being defined that includes the originator herself as the initial disseminator (*ID*) and a set of collaborating users (*Col*) as intended recipients, among which some are promoted as designated disseminators (*DD*). In particular, only *ID* and *DD* could distribute the information to other legitimate *Col*. The root of the sharing tree should always be an *ID* and the information flows coming from the *ID* belong to the type of dissemination. Similarly, the intermediate nodes should be *DD* and the information flows coming from *DD* belong to the type of re-dissemination. All other *Col*

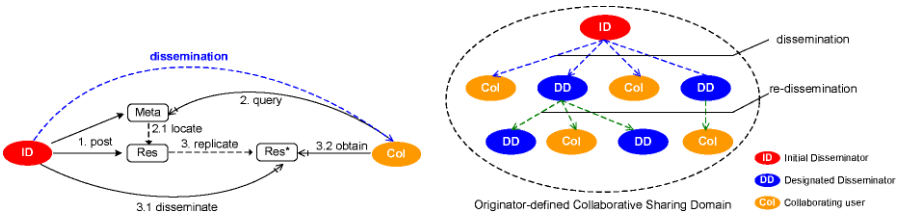


Fig. 1. Use patterns of resource dissemination and re-dissemination. (a) Resource dissemination. (b) Collaborative sharing pattern. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-2012-0446.)

inside the domain form the leaf nodes of the sharing tree. The link between each pair of nodes follows the same dissemination procedures as we identified in Fig. 1(a). The collaborative sharing domain is defined on per data resource basis to restrict the scope and flow of information dissemination within the collaboration community. And the originator needs to have a flexible yet effective way to include all unknown collaborating users within the domain and regulate their behaviors.

3. Role-based access management for ad-hoc resource sharing (RAMARS)

Role-based access control (RBAC) [43] provides great advantages in reducing the complexities of security management in large-scale enterprise-wide systems by using the notion of role to abstract users and privileges. In an ad-hoc collaboration environment, an originator could indirectly define her collaborative sharing domain and delegate information sharing capabilities to a set of roles, such as "data analyst" or "lab coordinator". By being assigned to these roles, collaborating users are automatically included in the originator's collaborative sharing domain and thus could obtain various capabilities to access or disseminate the data resource. Meanwhile, users are excluded from the collaboration if they are revoked from such roles. Therefore, bringing *role* in our framework becomes a natural choice to achieve the manageability for the originator.

Our role-based approach, however, distinguishes from traditional RBAC in terms of the role construct, permission-role assignment and user-role assignment. Existing RBAC models tend to rely on a single organizational policy to define roles within a physical administrative domain. We view roles as more flexible and more widely applicable to be defined independently across multiple administrative domains in a distributed environment. On the one hand, we design a set of *normative sharing roles* as global roles to abstract the generic activities for resource publication, discovery, access and dissemination. By defining these roles, a user's privileges for each step of information sharing can be authorized. Meanwhile, the intended behaviors between a normal collaborating user who is only supposed to access the data and a

designated disseminator who can further disseminate the data can be differentiated, so that the intended information flow among collaborators can be defined. On the other hand, an originator could define a set of *collaborator* roles as local roles to abstract her collaborative sharing domain for the particular data resource being shared within the community. For different data resources of the same originator, the originator could define different sets of collaborator roles. Through this approach, an originator could effectively tune the intended dissemination scope of different data resources for different collaboration purposes. As another extension to the traditional RBAC permission-role assignment, our framework allows an indirect permission assignment as *role reference*, where an originator could define the mapping between collaborator roles to global normative sharing roles when a collaborator role needs to exercise information sharing capabilities.

Introducing roles reduces the management complexity, yet the user-role assignment remains as an issue, as traditional RBAC models do not address unknown remote users and trust relevant aspects encountered in distributed collaborative settings. Our approach introduces another layer of abstraction, where users are automatically assigned to collaborator roles based on a finite set of assignment rules taken into consideration the attributes users own. Different from most role-based trust management approaches, credentials or certificates presented by users are not accepted at the same level of trust to testify the possession of attributes. Instead, a multi-level trust model is introduced for an originator to determine the trustworthiness of delegated authorities, thereafter to decide whether the claimant of such attributes can be accepted for the role assignment. A special type of *trust management constraint* with a set of evaluation policies and procedures has been introduced to determine the trustworthiness of user attributes given the supportive credentials. In doing so, we embed the trust-aware feature to the dynamic role assignment based on only trusted user attributes. Figure 2 summarizes all above discussions.

In [24], we proposed a Role-based Access Management for Ad-hoc Resource Sharing framework (RAMARS) to define the basic elements and relations that cover core features to be encompassed in collaborative information sharing systems. In particular, among all entities involved in ad-hoc collaboration, the collaborating users are the individual users who need to be authorized to gain access to the data. Each collaborating user is entitled with collections of attributes. A specific collection of attributes corresponds to one or more collaborator roles through a relation. A collaborating user must present credentials/certificates as proofs of the required attributes. Since we do not assume that an originator accepts all attributes claimed by a collaborating user, a special trust management constraint is in place to determine the trustworthiness of user attributes. The formal definition and relationships among elements are available in [24]. The details of *TM* constraint is discussed in Section 3.2. In terms of roles, RAMARS contains originator roles, collaborator roles, and normative sharing roles. The global operations for information sharing are assigned to normative sharing roles through a relation. Each collaborator role is mapped to one of these normative sharing roles accordingly in a relation. Finally in RAMARS, we

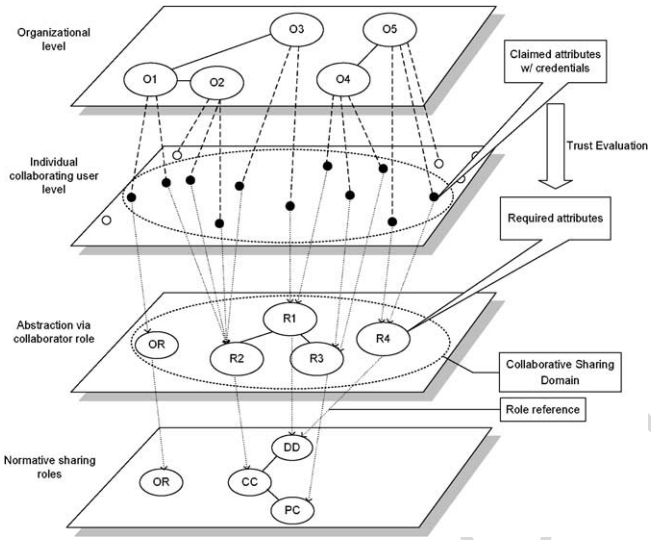


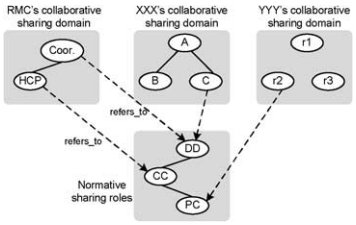
Fig. 2. RAMARS approach illustration.

use the term "physical" to distinguish the actual data resource from its metadata. This separation enables us to specify finer grained policies to protect the originator's resource. For example, in the process of data discovery, only metadata is shared with the collaborators.

3.1. An extended example

Before proceeding to introduce the trust-aware feature in trust management constraint, we discuss an extended example for the health surveillance scenario. The example serves for two purposes: to demonstrate how our framework introduced so far can be realized through the example; and to introduce a scenario with detailed trust management requirements that motivated our approach.

Assume in the collaborative sharing community, there are three normative sharing roles – *Designated Disseminator* role (*DD*), *Common Collaborator* role (*CC*) and *Potential Collaborator* role (*PC*) – being defined to associate with operations related to resource discovery, resource access and resource dissemination, respectively. And the role inheritance is defined as $DD \succeq CC \succeq PC$. *RMC* defines two collaborator roles – *Coordinator* role and Health Care Professional (*HCP*) role – as her collaborative sharing domain for the medical information, where *Coordinator* is a senior role mapped to *DD* role, and *HCP* is a junior role mapped to *CC* role. By this definition, all collaborating users must be assigned to either *Coordinator* role or *HCP* role in order to share *RMC*'s medical information. The members of *HCP* role can only access the data resource, while the members of *Coordinator* role may further re-disseminate the data to other legitimate users. To be assigned to *HCP* role,



(a)

HCP role: citizenship=US AND affiliation=ABC AND department=ECC AND status=on-duty

Required attributes	Credentials	Trust Propagation Graph
citizenship=US	Passport: asserted by US gov. Driver's license: asserted by DMV	RMC→US Gov.→Dave ↘ DMV ↗
affiliation=ABC department=ECC	Employment letter: asserted by AdminiStaff Outsourcing letter: delegate authority to AdminiStaff	RMC→ABC→AdminiStaff→Dave
status=on-duty	On-duty authorization: asserted by Dr. John Entitlement letter: delegate authority to Dr. John	RMC→ABC→John→Dave

(b)

Fig. 3. Collaborative sharing example. (a) Role mapping from collaborator roles to normative sharing roles. An originator (e.g., RMC, XXX or YYY) defines her own collaborator roles and maps them to the normative sharing roles for standard sharing operations. (b) Required attributes for *HCP* role assignment, supportive credentials provided by *Dave* and derived assertion paths.

RMC requires that the requester must be a US citizen according to security clearance guidelines, and he must be an on-duty staff in an Emergency Care Center (*ECC*) in the surveillance network. In addition, *RMC* allows the chair of *ECC* to be assigned to *Coordinator* role for data dissemination.

Suppose *ABC* hospital is an institute in the surveillance network and *RMC* delegates to *ABC* to assert its on-duty staffs in the *ECC*. Suppose *Dr. John* is in charge of the *ECC* in *ABC* and *Dave* is one of the on-duty physician assistants. In order for *Dave* to access the data, he must prove that he is a US citizen and is an on-duty employee in *ECC*. In particular, *Dave* could present both his passport and driver's license to prove his US citizenship. Suppose *ABC* has outsourced its human resource division to another company called *AdminiStaff*, *AdminiStaff* is responsible to assert *Dave*'s affiliation and department membership on behalf of *ABC*. And finally, *Dr. John* as the director of *ECC* could confirm that *Dave* is an on-duty physician assistant to share the information.

Figure 3(a) indicates the relationship between the local collaborator roles and the global normative sharing roles in the sharing network. In particular, originators, such as *RMC*, *XXX* and *YYY*, define their own collaborator roles and map them to the global normative sharing roles for standard sharing capabilities supported by the sharing infrastructure. Figure 3(b) shows an example of the required attributes for a user to be assigned to *HCP* role and the credentials provided by *Dave* to prove the possession of these attributes. Even though the credentials are compliance with the required attributes, they may not be trusted and accepted by the originator to guarantee the role assignment. We therefore introduce the *TM* constraint to evaluate the trustworthiness of user attributes given the supportive credentials.

3.2. RAMARS in trust management layer – TM constraint

Because a requester is assigned to collaborator roles based on his attributes, we begin with defining the user attribute first.

Definition 1 (Attribute). We let *ATTR* denote the set of attributes in the collaboration community. An attribute *attr* $\in$ *ATTR* is uniquely identified by its attribute name (`AName`). Each attribute name determines a specific domain for the values of the attribute, denoted as *domain*(*AName*). In particular, the value (*value*) of a user's attribute *attr* is a specific value within the attribute value domain or null, formally, *value* $\in$ *domain*(*AName*) $\cup$ {*null*}. In simplicity, we denote an attribute *attr* as an `(AName, value)` pair.

User attributes are asserted by credentials. We consider two types of credentials in our framework: *attribute credential* and *delegation credential*. An *attribute credential* is an attestation of one or more attributes issued to a user (holder) by an attribute authority (issuer). A *delegation credential*, on the other hand, is a statement specifying the delegation relationship between two attribute authorities to transfer assertion rights over certain attribute(s), where the delegator as the certifier and the delegatee as the credential holder. A credential is always associated certain validation constraints allowing anyone to ascertain if the credential is still valid. Normally, a validity period (t_1, t_2) is specified where a credential is valid from time t_1 to t_2, inclusive, and a Boolean delegation indicator is specified to restrict the further delegation of attribute authority. We use the notion of *Ctx* to abstract these constraints, and let Ctx_T denote the validity period and Ctx_D denote the delegation indicator.

Definition 2 (Credential). A credential is defined as a tuple of *cred* = (*holder*, *attrs*, *certifier*, *Ctx*), where *holder*, *certifier* $\in$ *EN*, *attrs* $\subseteq$ *ATTR* is the asserted user attributes, and *Ctx* is the validation constraints.

In particular, each credential *cred* = (*holder*, *attrs*, *certifier*, *Ctx*) can be represented as *certifier* $\rightarrow$ *holder* regarding the asserted attribute(s). For many of trust management approaches, the credential relation is modeled as a directed graph representing the trust propagation, and the authorization is granted when there exists a directed path from the source to the requester. In our scheme, we refer such path as an assertion path. The path discovery problem has been extensively investigated in graph theories and many trust management literatures [18,31]. We rely on the mathematical basis provided in these approaches to establish the trust propagation graph and derive the assertion paths given a set of credentials. Figure 3(b) shows the trust propagation graph for each required attribute given *Dave*'s credentials.

As we can see, a requester's attribute can be asserted by multiple paths and different requesters may present different sets of credentials for the same attribute. In our example, *Dave*'s passport and Driver's License establishes two assertion paths to assert his citizenship attribute. Another requester instead, say *Bob*, may present his birth certificate to prove the citizenship attribute. The trustworthiness of the attribute therefore should be varied according to these different assertion paths. Intuitively, we identify three major factors for an originator to determine the trust level of a user's attribute given different assertion paths: (1) the certifier(s) in an assertion path; (2) the

distance of an assertion path extended to a requester; (3) the number of (unfamiliar) certifiers asserting the same attributes as references. In particular, an originator may have various collaboration and trust relationships with different collaborating organizations. A trust relationship table thus is maintained by the originator for all collaborating organizations. And the attribute(s) asserted by different collaborating organizations result in different trust levels accordingly. For example, a high trust value may mean that the issuer of the attribute certificate has close collaboration relationship with the originator. In addition, as an assertion path is considered as a way for the originator to propagate trust to a remote user, the value of trust should decrease when aggregated along the path. Finally, an originator may consider to trust certain user attributes even when the certifiers are unfamiliar. This is often the case in recommendation-based systems [2] where the number of assertion paths are considered as an increased evidence on which the source bases its final trust value estimate. These factors are utilized by an originator to determine the trust level of the credentials as well as the assertion path composed of the credentials.

In our framework, we extend the trust propagation graph as a node-weighted trust graph $G = (V, E)$, where V is a finite set of distinguished entity nodes (i.e., attribute authorities and/or individual users) and E is a set of directed links/edges, each edge representing the credential relation from one node of the credential certifier to the other node of the credential holder. An originator defines the weight $w \in T$ for each vertex, where T is the trust level space $T = [0, 1]$. In particular, an originator explicitly specifies the trust value as weight for each known certifier nodes, and a default trust value is always specified for unfamiliar certifiers nodes. The node of originator is always trusted as 1 and the trust value for a collaborating user node is calculated along the established assertion path(s). The trust inference problem we deal with in our *TM* constraint can be formalized as:

Finding the trust value tv(N) that a source node M (the originator) should assign to the sink node N (the requester). Viewed as a generalized path problem, it is equivalent to finding the generalized distance between nodes M and N, given a weighted graph $G = (V, E, W)$ and $M, N \in V$.

Suppose $ap = (M, v_1, v_2, \ldots, v_k, N)$ is an assertion path in G, we see the trust value decreases when aggregated along the path. Therefore the trust value for the requester N is calculated based on the overall weight of the assertion path $w(ap)$:

$$tv(N) = w(ap) = w(v_1) \times w(v_2) \times \cdots \times w(v_k). \tag{1}$$

When there are multiple assertion paths available, we see it as an increased evidence for the trust evaluation. Therefore, suppose $ap_1, \ldots, ap_k$ are k paths established by the credentials, the trust value $tv(N)$ is calculated as a function of all these paths:

$$tv(N) = \sum_{i=1}^{k} w(ap_i). \tag{2}$$

```
CalculateTrustValue(Graph G, Weight weight, Vertex source, Vertex sink){
    // find all simple paths from source to sink
    Set Apaths;
    LinkedList visited.add(source);
    DepthFirst(G,visited);

    // calculate the trust value
    TrustValue = 0;
    PathWeight = 1;
    FOR each path in Apaths {
        // calculate path weight
        FOR each node in path {
            IF(node≠sink) // calculate path weight
                PathWeight = PathWeight * weight(node);
        }
        // calculate trust value combining all path weights
        TrustValue = TrustValue + PathWeight;     }
    return TrustValue;
}

DepthFirst(Graph graph, LinkedList visited) {
    LinkedList nodes = graph.adjacentNodes(visited.getLast());
    // examine adjacent nodes
    FOR each node in nodes {
        IF (visited.contains(node)) {
            continue;
        }
        IF (node=sink) {
            visited.add(node);
            Apaths.add(visited);
            visited.removeLast();
            break;
        }
    }
    // in depth-first, recursion comes after visiting adjacent nodes
    FOR each node in nodes {
        IF (visited.contains(node) || (node=sink)) {
            continue;
        }
        visited.addLast(node);
        DepthFirst(graph, visited);
        visited.removeLast();
    }
}
```

Fig. 4. Trust value calculation algorithm.

Given a weighted trust graph, we design an algorithm as shown in Fig. 4 to calculate the trust value from the source vertex (originator) to the sink vertex (requester). Thealgorithm first implements a recursive depth-first search (DFS) algorithm to find all non-cyclical assertion paths from the source to the sink. In particular, the algorithm uses a linked list, called *visited*, to keep track of the set of nodes to be explored. *visited* is initialized with the source. The algorithm progresses by searching the immediate adjacent vertex to the last node in *visited* and going deeper and deeper until the sink node is found, which means a path is found from the source to the sink. And such path, as maintained in *visited*, is added to *Apaths* to keep track of the paths being found. Then the search backtracks to the previous node it has not finished exploring in *visited* and begins another round of iteration. After all paths are found, the algorithm calculates the weight for each assertion path as maintained in *Apaths* and combines all the paths to calculate the trust value from the source to the sink as indicated in Eqs (1) and (2).

As we can see, the crucial parameter affecting the complexity of the algorithm is the number of paths from the source to the sink. It is referred as an "all simple paths" problem [36]. Theoretically, the computation complexity for a DFS is well known as $O(|V|+|E|)$, linear in the size of the graph. However, as indicated in [35], the worst case for "all simple paths problem" may be exponential to the size and structure of the graph. Given the graph G and any of the assertion paths in G, we use d to denote maximum depth of the assertion path from vertex M to N, and we define b as the branching factor [49], or the maximum number of neighbors of a vertex v_i. The complexity of the all simple paths algorithm can be computed as following: for each vertex adjacent to the last one in the path (b), it does a linear scan over the linked list of previously visited nodes (d). This results in $O(bd)$ steps of computation. And the computation recursively computes for d times along the path, resulting in the overall complexity of $O((bd)^d)$. In our trust evaluation, the number of supportive credentials provided by a requester for a given attribute would not be large, say 10. We believe the algorithm could achieve efficient performance with such a low computation scale. However, when the size of the graph does grow to a higher degree, instead of finding and combining all assertion paths for the trust value, we may convert the trust inference problem to:

Finding an assertion path including a sequence of nodes $(M, v_1, v_2, \ldots, v_k, N)$ *that has the highest trust value among all assertion paths starting at* M *and ending at* N.

This is motivated from the common sense observation that the trust evaluation needs to efficiently derive a most trusted assertion path as the evidence of the claimed attribute. Mathematically, we reduce the original "all simple paths" problem to "single-source shortest path" problem in a weighted directed graph, which could be efficiently solved by Dijkstra's algorithm [37]. We leave the implementation of this algorithm as our future work.

Given the trust values for the requester's attributes, an originator may finally specify a threshold to determine whether to trust the attributes for the role assignment. As a summary, our *TM* constraint consists of the following evaluation procedures:

(1) Credential processing: validate all credentials based on their context constraints. Build the trust graph with valid credentials.
(2) Trust value determination: calculate the trust value for the requester's attribute according to the established trust graph and the weight table specified by the originator.
(3) Trustworthiness assessment: determine the trustworthiness of the requester's attribute according to the originator's threshold.

4. Policy specification

As a policy-driven approach, we now present the policy specification to realize the components conveyed in RAMARS model. From the perspective of originator-control, an originator defines collaborator roles as her collaborative sharing domain

and delegates access and dissemination capabilities to the roles through role reference. The originator also specifies the policies to govern the user to collaborator role assignment in terms of required attributes, while the trust evaluation rules are also specified. In our policy framework, all these policies are specified inside the Role-based Originator Authorization policy set (**ROA**). ROA policy set is the major policy component to make the originator in control of her information. As information may be disseminated among collaborating users, the system should guarantee the originator's ROA policies being enforced correspondingly along with the information dissemination. The traditional "sticky policy" paradigm [28,51] is not efficient in transferring the whole ROA policy set at each time of the dissemination, and it causes the policy synchronization concern as the originator needs to adjust her ROA policies from time to time. In order to facilitate distributed deployment and enforcement of the ROA policy set in a more efficient way, we design a Root Meta Policy Set (**RMPS**) to declare the ownership of a particular resource and associate the location of ROA policies. By doing this, an originator can modify her ROA policies locally, and a lightweight RMPS policy propagates among collaborating users to locate the ROA policies. In addition, different resources belonging to the same originator could share the same set of ROA policies, so that policy reusability and portability could be achieved. On the collaborator's side, the validation constraints for each credential specifies the rules to restrict the validity of the credential, which requires clear syntax and semantics defined in **CREDPolicy**. The policy specification, as shown in Appendix, follows the same notion of terminals and nonterminals as defined in the ISO standard for extended Backus–Naur form (EBNF) [1].

4.1. ROA policy set

The ROA policy set is the major component for an originator to express the authorization as well as trust management policies for the data resource being shared. ROA policy set contains the following policy elements:

Role Policy Set (RPS) is a role specification policy and each role is defined as a single Role Policy (*RP*) element. A *RP* contains an optional policy `id` and a `RoleName` as the unique role identifier. There are two types of roles that an originator could define, the normative sharing role (`N`) and the collaborator role (`C`), respectively. Each role is associated with a specific Capability Policy (*CapPolicy*) that contains the capabilities being assigned to the role. Therefore the permission-role assignment relation is achieved through the reference of `CapPolicyId`. In our specification, elements such as policy `id` and `RoleName` are specified as "*?arbitrary string?*", which means the originator has freedom to name these elements.

Capability Policy Set (CPS) specifies the actual capabilities assigned to each role. In a *CapPolicy*, `CapPolicyId` serves as a unique identifier that is referenced from the corresponding role specification policy. As discussed in our

RAMARS model, the normative sharing roles abstract specific sharing operations, and the collaborator roles obtain the capabilities through mapping to one of the normative sharing roles. Therefore, for the "N" type of roles, the capability policy defines a set of operations being assigned to the role. And for the "C" type of roles, it references to the mapped "N" type role through `CapPolicyId_MappedRole`. With this reference, all collaborator roles can be eventually led to the actual authorized sharing operations. In addition, the role hierarchy relationship is captured in a similar way through policy references from the senior role to its junior roles. We directly represent role mapping and role hierarchy as capability set references, since the authorization consequence of role mapping and role hierarchy is the capability aggregation, where the target role is capable to exercise all permissions being assigned to the mapped roles and its junior roles. By doing this, we are able to simplify the policy structure and reduce the number of potential policies to be evaluated. Another important feature worth mentioning here is that only the authorized operations are defined in each *CapPolicy*, while the target object (data resource) of the operation is defined in a separate policy – *RMPS*. This is different from the normal permission definition as operations towards objects. We apply this special design for two reasons. On the one hand, the sharing operations are identified for all sharing infrastructures to support regardless of the specific resources being shared. On the other hand, by separating the resource from the operations, the capability policies can be re-used where multiple resources could share the same set of operation policies.

Role Assignment Policy Set (RAPS) contains one or more sub-policies (*RAPolicy*) to define the required attributes for each collaborator role. The collaborator role can be assigned according to different sets of required attributes, each denoted as a `ReqAttrGroup`. A `ReqAttrGroup` defines a set of attribute predicates mapping the values of the user's attributes to the required attributes with expected values. In particular, each required attribute is identified by the `Aname`, and the `TargetValue` specifies the expected attribute value within *domain*(*Aname*) that is to be checked against the user's attribute value based on the comparison operator `ComparisonOP`. A Boolean value is returned if the attribute value satisfies the predicate. For instance, in expressing an attribute predicate that requires "*a user must be a US citizen*", the `Aname` is "`citizenship`", and the `TargetValue` is "`US`" with the `ComparisonOP` as "=". If a user has an attribute of `(citizenship,US)`, then a Boolean value *true* is returned according to this predicate. In order to aggregate the Boolean values for all specified predicates, a `CombinationOP` is in place to specify the logical combination rule: (i) "AND" implies that all attribute condition predicates must be *true*; (ii) "OR" implies that at least one attribute condition predicate must be *true*; and (iii) "NOT" implies that none of the attribute condition must be *true*. The role is assigned after a *true* value is eventually derived from the `ReqAttrGroup` evaluation.

Trust Assessment Policy (*TAP*) is used in two types of functionalities achieved by *TM* constraint: (i) to determine the trust value of the requester's claimed attributes; and (ii) to make the trust decision on the claimed attributes. We realize the functionalities using two policy elements referred to as *TAP.TL* and *TAP.TD*, respectively. *TAP.TL* is used for an originator to establish the trust relationship table for each collaborating organization. In particular, *TAP.TL* defines the target `Attribute(s)` under evaluation, the delegated collaborating organization (`Certifier`) and the trust value (`TrustValue`) as a real number in [0, 1] to be assigned. Each attribute is represented as an `Aname` and an optional `Avalue`, which gives an originator more flexibility to define the attribute in a general way to specify the attribute name only, or in a specific way to narrow down the value of the attribute. *TAP.TD* policy, on the other hand, specifies the threshold of the trust level for the claimed attributes to be trusted. *TAP.TD* is specified as one or more `Attributes` to be evaluated including the `Threshold` and the `ComparisonOP`. As the trust level is defined as a partial order, we assume the default `ComparisonOP` is "$\geqslant$". The evaluation returns *true* when the trust level of the attributes is equal or greater than the specified threshold. By returning *true*, the attributes are trusted and can be promoted for the role assignment evaluation.

4.2. Root meta policy set (*RMPS*)

As *ROA* policies are specified independent from the applied data resource and can be deployed in an originator's local domain, *RMPS* is designed to associate *ROA* policies with the specific `Resource` and enable the policy enforcement system to locate *ROA* polices. In *RMPS*, the `Resource` is represented by a URI conforming to RFC2396 [11], and originators are identified by `OriginatorId` elements in the format of X.500 distinguished names (DNs) [48]. The originator's *ROA* policies are referenced through the `ROAPolicyLocation` URI.

4.3. Credential policy (*CREDPolicy*)

CREDPolicy specifies the validation constraints for a credential. Besides the elements of `Holder`, `Certifier` and asserted/delegated `Attributes`, each *CREDPolicy* contains one or more `Context` constraint predicates. And a logical combination operator `CombinationOP` is specified to aggregate the Boolean evaluation values for each `Context` constraint predicate. In particular, the validity period constraint `ValidityPeriod` is defined with `Start` date and `End` date. To evaluate, an environmental parameter "*CurrentDate*" should be compared, a *true* value is derived when the current date is in between the period of `Start` and `End`. In terms of the `Delegation` constraint, a simple Boolean indicator is defined to restrict the further delegation. A credential is valid when a *true* value is derived from all `Context` constraints.

4.4. Policy example and policy evaluation

The OASIS standard for XACML [39,40] has been well adopted as a general policy language used to protect resources as well as an access decision language. In supporting RBAC, OASIS has recommended a specification for RBAC policies [38] ("RB-XACML" for simplicity). Inside RB-XACML, roles are specified in *Role PolicySet*, permissions are defined in *Permission PolicySet*, and user-role assignment are defined in *Role Assignment PolicySet*. With XACML 3.0, the delegation chain can be derived through policy reduction back to the original delegator's policy. Integrating these features and accommodating the standard syntax, we design a set of XACML-based policies as an implementation of our proposed RAMARS policy framework. Figure 5 illustrates the policy examples based on the scenario in Section 3.1.

Figure 5(a) shows an overview of the policy structure for *RMPS* and *ROA*. *RMPS* specifies that the `Resource` "*file:///usr/data*" has "*CN = RMC*" as the originator, where *RMC*'s *ROA* policy set is located at "*ldap://rmc.com:389/o=RMC*". In *RMC*'s *ROA* policy set, *RPS* defines two collaborator roles as *Coordinator* and *HCP*, and normative sharing roles as *DD*, *CC* and *PC*. *CPS* specifies the capabilities associated with these roles. Utilizing the policy reference, the role hierarchy is represented as *Coordinator* $\succeq$ *HCP* and *DD* $\succeq$ *CC* $\succeq$ *PC*. The role mapping is carried out in similar ways where the *Coordinator* and *HCP* roles are mapped to *DD* role and *CC* role, respectively. Figure 5(b) shows the detailed definitions in the policy sets.

In terms of role assignment, *RAPS* specifies an example of the attribute requirements for *HCP* role. *HCP* role requires the following attributes: `citizenship=US AND affiliation=ABC AND department=ECC AND status=on-duty`. In order to evaluate the trustworthiness of user attributes, *RMC* also defines the trust assessment policies. The *TAP.TL PolicySet* example in Fig. 5(c) specifies that "`CN=ABC`" is assigned with trust value of 1 for asserting the attribute of `(affiliation,ABC)`, while other entities are assigned with trust value of 0.5. And the *TAP.TD* policy example specifies that the attribute is trusted when the trust level is *larger or equal to 0.5*.

The credential policies (*CREDPolicy*) defined by credential certifiers must have the `PolicyIssuer` element to indicate the credential issuer. To differentiate the two types of credentials, we specify attribute credential policy (*ATTR Policy*) and delegation credential policy (*DLGT Policy*) separately. Figure 5(d) shows examples in both types and the process of policy reduction for the assertion path. In particular, *ATTR Policy* specifies a credential issued by `AdminiStaff` to assert `Dave`'s affiliation attribute. *DLGT Policy* specifies that "`CN=ABC`" delegates to "`CN=AdminiStaff`" the right over `(affiliation,ABC)` attribute between the period of `1/1/2009` and `12/31/2009`. Besides, `ABC` also specifies `dlgt=false` as the delegation constraint to restrict further delegations by `AdminiStaff`. In order to evaluate the validity of these credentials. An attribute validation query is generated to query whether *Dave* is authorized for the `(affiliation, ABC)` attribute on *current date* (*6/1/2009*). According to *ATTR Policy*, a `Permit`

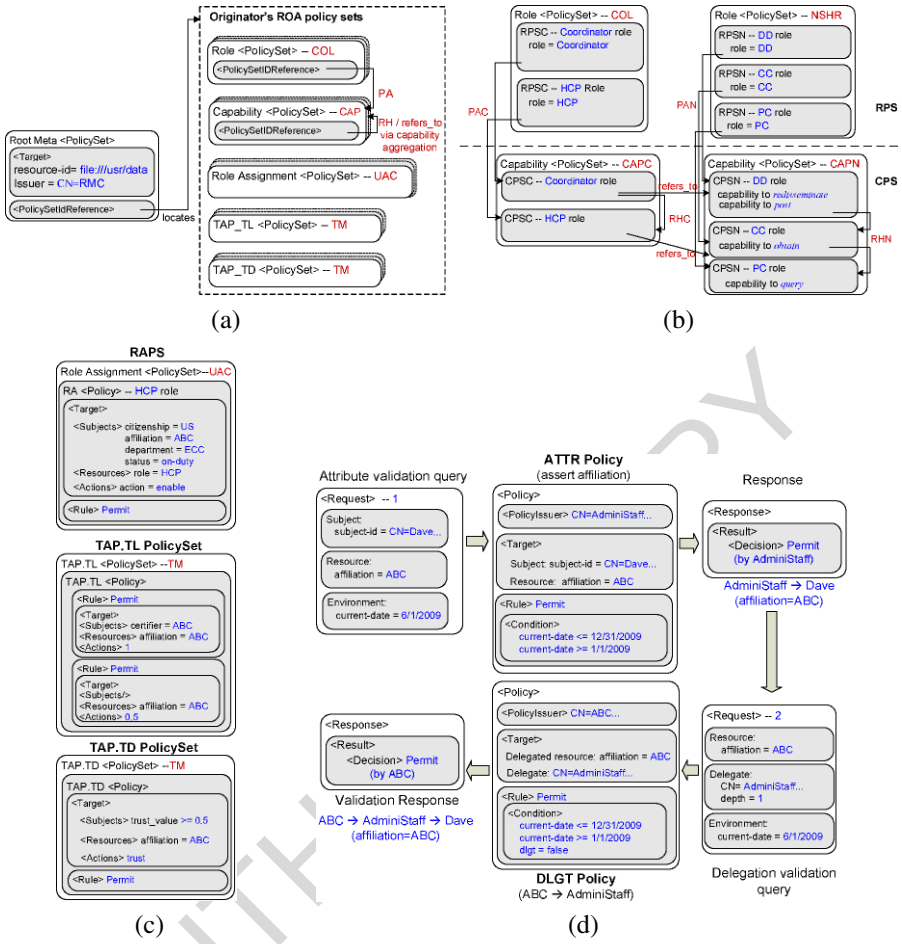


Fig. 5. XACML policy examples. (a) RMPS–ROA overview. (b) RPS–CPS details. (c) RAPS–TAP details. (d) CREDPoicy examples and assertion path validation. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-2012-0446.)

decision is derived. Since *ATTR Policy* is issued by `AdminiStaff`, the `Permit` decision needs to be further evaluated to make sure that `AdminiStaff` is an authorized/trusted certifier to assert the attribute. A delegation validation query is then generated for such purpose. According to *DLGT Policy*, another `Permit` decision is derived after the evaluation. Thus a valid assertion path is derived as `ABC` → `AdminiStaff` → `Dave`.

Having all policies being specified, now we are ready to discuss the policy evaluation procedures. In our system, we introduce the *Context Handler* to conduct a series of XACML request-generation and decision-processing operations. We now focus on the evaluation process, and leave the details of system components to next

section. As an example, we try to evaluate whether *Dave* is allowed to *obtain* the data file (*file:///usr/data*) with all six credentials as listed in Fig. 3.

Pre-processing: The evaluation process is triggered when the policy evaluation engine receives *Dave*'s file access request with supportive credentials. Since credentials may assert different user attributes, a preprocessing operation is conducted to derive the assertion paths according to their asserted attributes. In XACML evaluation environment, credentials are parsed and grouped based on same *Resource* and *Delegated resource* attributes. In our example, *Dave*'s six credentials are categorized into four groups asserting the attributes of `citizenship`, `affiliation`, `department` and `status`.

Trust evaluation: The first step of trust evaluation is to derive the assertion paths for the asserted attribute. This is realized in XACML through the process of policy reduction as shown in Fig. 5(d). An assertion path is derived as `ABC` $\rightarrow$ `AdminiStaff` $\rightarrow$ `Dave`. The next step of evaluation is to determine the trust value of the asserted attributes based on the assertion path. According to *TAP.TL* in Fig. 5(c), the trust value based on the assertion path is calculated as $tv(Dave) = 1 \times 0.5 = 0.5$. Finally, the system decides on the trustworthiness of the attribute based on the achieved trust value. According to the example *TAP.TD* policy, a trust value of 0.5 meets the minimum required trust threshold, so that the attribute of `(affiliation,ABC)` is trusted for the role assignment.

Role assignment evaluation: After the trust evaluation, all trusted user attributes are promoted. For each role in the system, the *Context Handler* generates a role assignment request. The *RAPS* policy is evaluated and the user is assigned to the role when a `Permit` decision is derived. In our example, if *Dave*'s attributes of `citizenship`, `affiliation`, `membership` and `status` are all trusted, the *HCP* role is eventually assigned to *Dave*.

Authorization evaluation: Finally, the PDP conducts the authorization evaluation by evaluating *RMPS*, *RPS* and *CPS* to examine whether the assigned *HCP* role is allowed to conduct the "*obtain*" action on the file resource (*file:///usr/data*). As the role of *HCP* is mapped to *CC* role and *CC* role is allowed to *obtain* the file, *HCP* role is then allowed to *obtain* the file. The final decision `Permit` is derived, and thus *Dave*'s access is granted.

5. System design and prototype implementation

As part of our on-going research efforts, we have designed and implemented a prototype system, called *ShareEnabler*, to demonstrate how the proposed RAMARS framework and policy specification can be realized as comprehensive authorization services within the context of collaborative sharing applications.

ShareEnabler adopts a specific communication infrastructure from a P2P based scientific information sharing toolkit SciShare [10] developed by Lawrence Berkeley National Laboratory. In our collaborative sharing system, each participating entity is represented by a ShareEnabler agent that executes sharing services on its behalf. The originator uses the agent to post file resources and share those resources with other collaborators, and collaborators operate on their agents to query and download files. Similar to existing P2P file sharing systems, the resource discovery involves broadcasting a query to all known peers, while sending response and disseminating files are bound to unicasting communications. Figure 6 shows an overview of the system infrastructure. Suppose the collaborative sharing group consists of six peer participants, each participant is represented as a ShareEnabler agent. Agent 1 broadcasts a query message to all known peers in the group (step 5). Upon receiving the query message, Agents 2–5 look up their own posted contents. Agent 2 finds the matched content(s), evaluates the originator's *ROA* policies and sends a unicast query response with the metadata of the authorized content(s) back to Agent 1 (steps 1′–13′ in Agent 2), while Agents 3–5 are not necessary to respond to the requester. We call this process as metadata sharing. Agent 1 then can send a download request to Agent 2, and Agent 2 further checks with the originator's *ROA* policies and initiates the data transferring process if the requester is authorized to download the file. We call this process as data sharing. The access control for sharing of both metadata and real file data is carried out by our proposed RAMARS framework.

Figure 6 also shows the system components inside a ShareEnabler Agent and their interactions in the process of metadata sharing between ShareEnabler Agent 1 as the

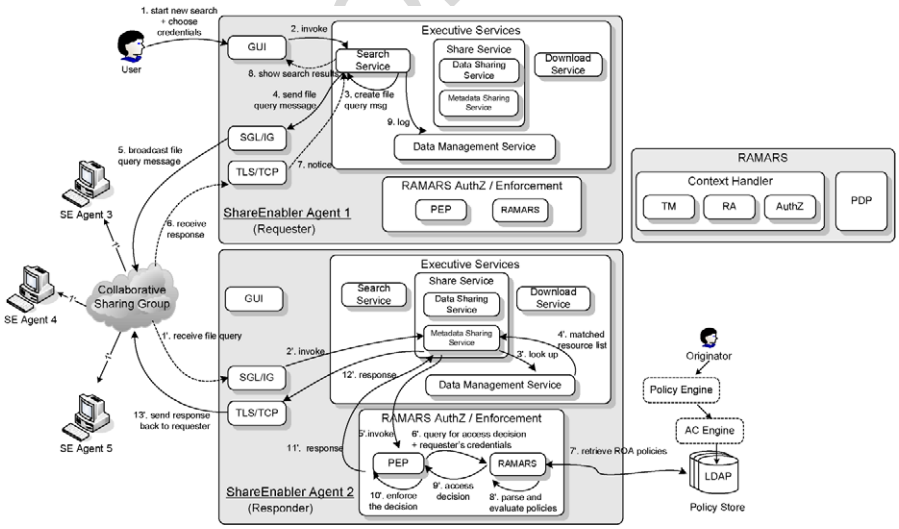


Fig. 6. ShareEnabler system infrastructure and architecture. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-2012-0446.)

requester and Agent 2 as the responder. Each ShareEnabler agent is composed of five components: Graphical User Interface (GUI), Executive Services, RAMARS AuthZ/Enforcement service, Secure Group Layer/InterGroup protocol (SGL/IG) [4, 9] and Transport Layer Security/Transmission Control Protocol (TLS/TCP). GUI is the interface through which a user operates and executes sharing services. Executive Services are the real services required by collaborative sharing behaviors including Search, Download and Share Services. All these services interact with an underlying Data Management Service as the backend database in the agent. RAMARS AuthZ/Enforcement service is the central component that conveys our proposed RAMARS framework for the core access and dissemination control services. The PEP is responsible for the request processing and access decision enforcement. The RAMARS Engine, which consists of the Context Handler and the core PDP Engine, is designed for the policy retrieval and all policy related evaluations as we discussed in Section 4.4. The secure and reliable multicast communication is achieved by the combination of SGL/IG protocols, where a group session key is established among all authenticated peers to encrypt the communication messages. Meanwhile, the unicast communication security is achieved by TLS when peers play traditional roles of client and server.

On the requester's side (ShareEnabler Agent 1) in metadata sharing, a user interacts with the GUI to specify the query criteria and choose his credentials[1] (step 1). GUI invokes the Search Service to formulate and embed the user's credentials into a query message. The query is then broadcasted to all peers in the collaborative sharing group through the SGL/IG secure channel (steps 2–5). Upon receiving responses from other peers, TLS/TCP notices the Search Service with the response messages (steps 6 and 7) and shown in the GUI (step 8). The search results are logged into Data Management Service (step 9).

On the responder agent side (ShareEnabler Agent 2), the SGL/IG module notices the Metadata Sharing Service (steps $1'$ and $2'$) upon receiving the query message. The Metadata Sharing Service separates the content request and the requester's credentials from the query. The Data Management Service is then invoked to find matched resources against the content request (step $3'$). For each file posted in the agent, the originator defines her *ROA* policies and stores in a local LDAP policy repository using a facility policy generation tool, while the root policy *RMPS* is always attached with the file in the agent. The Data Management Service returns a list of matched files along with their associated *RMPS* policies to the Metadata Sharing Service, through which the PEP is invoked for access checking and enforcement (steps $4'$ and $5'$). The PEP generates an access request and forwards the requester's credentials to the RAMARS Engine for the access decision evaluation (step $6'$). According to the information specified in *RMPS*, the RAMARS Engine retrieves *ROA*

[1]Here we assume a user's credentials are locally stored for the demonstration purpose of the authorization service. Our system could be extended to support credential retrievals from online credential providers.

policies from the originator's LDAP policy repository and examine whether the requester is allowed to *query* the file (steps 7′–9′). Upon receiving the access decision from the RAMARS Engine, PEP enforces the decision by only returning the authorized files to the Metadata Sharing Service (steps 10′ and 11′). Finally, the Metadata Sharing Service formulates the response message and sends back to the requester through TLS/TCP protocol (steps 12′ and 13′).

As we have shown above, *ROA* policies are deployed separately from the major application and enforcement components. These policies are pulled at runtime when the RAMARS Engine in the ShareEnabler agent needs to make an authorization decision. Therefore, an originator can easily maintain and change the policies without requiring changes to the sharing service systems. We decide to apply X.509 attribute certificates to encapsulate access management policies. X.509 attribute certificate (AC) is a basic data structure in Privilege Management Infrastructure (PMI) [22] to bind a set of attributes to its holder. With its portability and flexibility, AC is considered as an ideal container for subject attributes as well as authorization policies in ShareEnabler. We also developed a separate facility application, called Administrative Policy Editor, for an originator to create *ROA* policies, generate policy ACs and store in the LDAP policy repository. In terms of a user's credentials, X.509 public key certificate (X.509 PKC) is the major identity credential required by the P2P file sharing infrastructure for each peer agent to authenticate itself to other agents for secure communications. The certificate can be either self-signed or signed by a trusted certificate authority. The self-signed certificate could be used by a new peer (called pseudo user) to join the community quickly [10]. However, the X.509 PKC is not the major credential to determine a user's privileges. Instead, the user's privileges are determined through the attributes he possesses, which are bound into X.509 attribute certificates. We have demonstrated how XACML policies can be utilized to specify credential policies and how the credentials could be validated through XACML policy evaluation mechanisms. To make the implementation consistent, we directly embed the XACML *CREDPolicy* as attributes in ACs to be transferred between peer agents for authorization purposes.

In terms of dissemination control mechanisms, the goal of access and dissemination control for ShareEnabler is to guarantee the file resource to be distributed within the collaborative sharing domain defined by *ROA* policies. Our system applies a distributed policy propagation and enforcement scheme with self-enforcing and self-monitoring features at each ShareEnabler agent level. As demonstrated in the previous section, each ShareEnabler agent ensures that *ROA* policies are enforced locally by the RAMARS AuthZ/Enforcement component, so that only legitimate peers can obtain the requested file. In addition, *ROA* policies should be propagated and enforced by recipient ShareEnabler agents as well when they act as disseminators to respond requests from other peers. Since *RMPS* plays an important role for the ShareEnabler Agent to locate and enforce an originator's *ROA* policies. It is essential to make sure *RMPS* is propagated along with the file dissemination, and the confidentiality and integrity are properly protected when it leaves the originator's

domain. In order to achieve these requirements, we introduce a new self-contained cryptographic data structure, called *SEFile*, to encapsulate the original data file with the associated *RMPS*. When an originator tries to post a file in her ShareEnabler agent, the agent creates the SEFile from the original data file and encrypts it with a predefined secrete key. The encrypted SEFile is then shared with all other collaborators through the originator's ShareEnabler agent. Therefore, instead of receiving the raw data file, the collaborators receive the encrypted SEFiles. The SEFile can only be decrypted at runtime when the receiver uses a particular SEFile parser associated with ShareEnabler agent. By doing this, we empower the ShareEnabler agent to be extensible for more advanced dissemination control and tracking mechanisms. To prevent unauthorized dissemination, when a user tries to *post* a pre-obtained data file in his ShareEnabler agent for re-distribution purpose. The SEFile is detected and decrypted by the PEP to get the original *RMPS*. And RAMARS Engine is prompted to make decision on whether the user is authorized to *post* and *redisseminate* the resource. In other words, the ShareEnabler agent would check whether the user is a legitimate designated disseminator in every re-dissemination attempt. PEP declines the user's *post* request if he is not authorized to do so. Otherwise, ShareEnabler posts the resource in the SEFile format automatically and allows it to be re-disseminated.

We have implemented a prototype ShareEnabler system using Java. In our prototype, we use JDK1.5 core packages as well as other necessary libraries to develop components specified in the system architecture. Especially, we adopt SciShare's Reliable and Secure Group Communication (RSGC) package for the implementation of SGL/TLS communication protocol as well as the basic authentication mechanisms underneath. We extend SICS's XACML3.0 implementation [46] to accommodate the functionalities utilized in XACML policy evaluations. A special path tracker has been embedded in the implementation to record the credential policy evaluation paths so that the valid assertion paths can be captured. In addition, IAIK's Java crypto library is used toimplement major cryptographic and X.509 attribute certificate related modules. Finally, the IPlanet Directory Server serves as the back-end LDAP policy repository. The proof-of-concept implementation of ShareEnabler system has been completed for further testings and evaluation. Figure 7(a) shows an interface when a user *Dave* searches a particular resource. For the responding agent, only the data resources that *Dave* is authorized to *query* are returned back and shown in *Dave*'s agent. A sample authorization message on the responder side is also shown under the interface. Figure 7(b) shows an interface of Administrative Policy Editor for the originator *RMC* to create *ROA* policy attribute certificate. All *ROA* policies are encoded as attributes in an X.509 AC.

6. Performance evaluation

The purpose of performance evaluation is to examine the scalability as well as efficiency of our ShareEnabler system. In particular, we evaluate how well the RAMARS Engine as the authorization module scales along with the increased evaluation complexity and also analyze the overhead that RAMARS Engine has put on

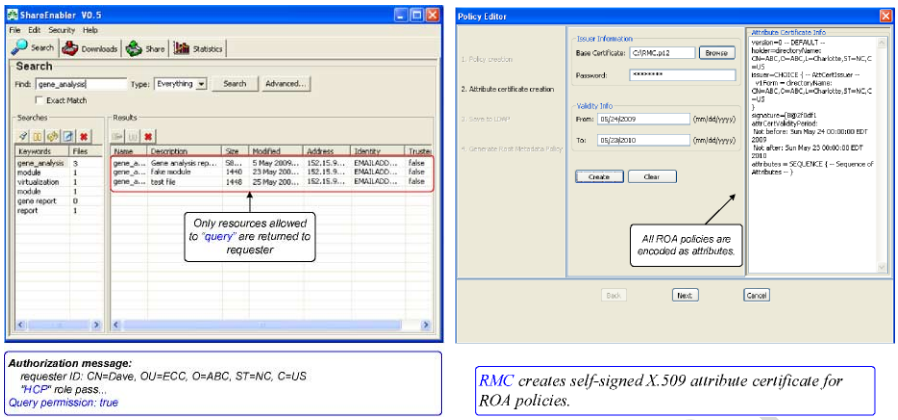
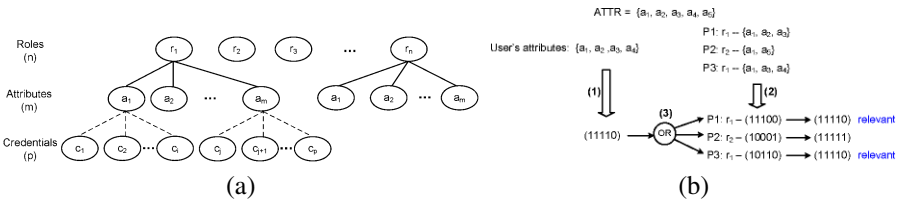


Fig. 7. User interfaces of ShareEnabler agent and administrative policy editor. (a) New search and search results. (b) Attribute certificate creation. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-2012-0446.)

the underlying P2P based scientific file sharing infrastructure where ShareEnabler is built upon. To examine the scalability of RAMARS Engine, each procedure of the policy evaluation, such as trust evaluation (TM evaluation), role assignment evaluation (RA evaluation) and authorization evaluation (AuthZ evaluation), is measured when we change the number of attributes, credentials and roles involved in the evaluation. As we discussed earlier, RAMARS Engine takes an access request along with a set of supportive credentials in X.509 attribute certificate format as an input, and returns a Boolean valued decision indicating whether the access is granted or not.

We have developed tools to generate and monitor workloads. Monitors are embedded in RAMARS Engine to measure the time spent for each evaluation step. The total time consumption of RAMARS Engine is also measured between the acceptance of request and the return of final decision. In our workload generation, all roles and attributes are uniformly created to evaluate our prototype. Those roles are randomly created without hierarchies but associated attributes implicitly indicate relationships among roles. The assignment of each role requires the same set of attributes and the trust evaluation for each attribute requires the same level of trust for its supportive assertion paths. The number of credentials are varied by increasing the depth of a delegation path. Figure 8(a) shows a general testcase setting for the relationships of roles, attributes and credentials. In particular, there are n roles involved in the system, and a user must possess m attributes to be assigned to each role. In supporting the claims of m attributes, a total of p credentials are presented with p/m credentials for each attribute claim. To claim m attributes, it requires at least $p = m$ credentials (one credential for each attribute claim). When there is one step of delegation, each attribute is supported by an assertion path with the depth of 2, which then implies



(a) (b)

Fig. 8. Performance testing settings. (a) RAMARS engine performance testcase setting. (b) Bitmap policy indexing setting. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-2012-0446.)

$p = 2m$ credentials in total. Therefore, any incremental change of delegation depth results in an increase of the number of credentials. The experiments ran on a Pentium M with 512 MB RAM running at 1.8 GHz with Windows XP. Time consumption for each evaluation process is measured in milliseconds and calculated as the average of 100 independent test runs.[2]

Test 1 – *Credential increase and credential indexing*

In collaborative environments, we expect the number of roles and attributes involved in the authorization are relatively stable, while a user's submitted credentials may vary dramatically. In this sense, the performance of RAMARS Engine handling credentials is a major factor to the overall scalability of the system in practice. Our first experiment is conducted by fixing the number of roles and attributes, but increasing the number of credentials. Figure 9(a) indicates the initial testing result when there are 10 roles and 10 attributes, as the number of credentials gradually increases from 10 to 80. This initial result indicates that the time spent in TM evaluation grows fast along with the increase in the number of credentials. We revisit our implementation of TM evaluation and observe that the pooling of all credentials in TM evaluation puts heavy burdens on the evaluation engine, as it must scan all credentials in each step of trust evaluation. For such a root cause, we implement an indexing mechanism based on a Map data structure. In particular, as a Map structure associates *keys* with *values*, we use the claimed attribute as the key and a set of relevant credentials as the value. For instance, when there are p credentials asserting m attributes. The credential index then keeps m entries of records, where each record maintains the attribute as the key and p/m credentials as the value. The trust evaluation engine only takes one entry of `(attribute, credentials)` at each time for evaluation so that the amount of credentials to be scanned could be reduced significantly. Compared to the result of our initial test, as shown in Fig. 9(b), the indexing mechanism demonstrates dramatic advantages. When there are 80 credentials, the total evaluation time for RAMARS Engine is reduced from nearly 3.5 s to less than 2 s with a flatter increase trend. The time consumption of TM evaluation

[2]For brevity, our performance analysis omitted the network-related overhead.

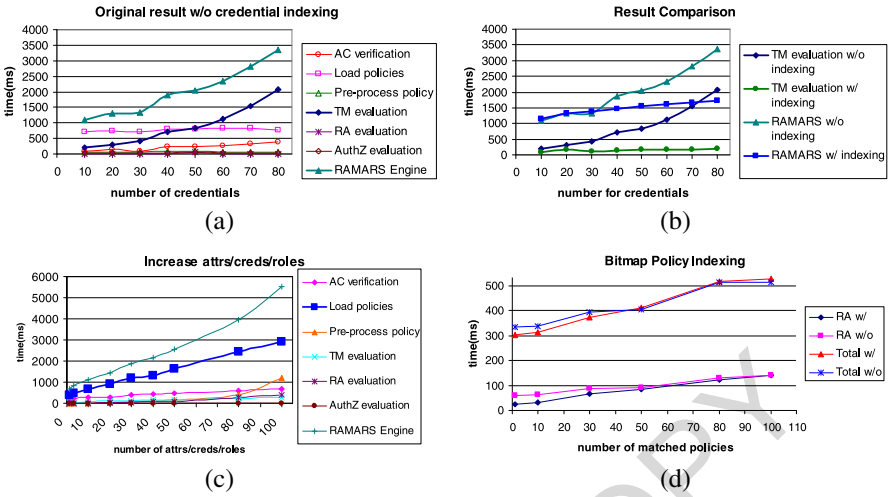


Fig. 9. Performance testing results. (a) Result without credential indexing. (b) Comparison with credential indexing. (c) Increasing roles and attributes. (d) Comparison with bitmap policy indexing. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/JCS-2012-0446.)

stays almost stable at around 0.3 s. This is a desirable property and has made the system more scalable to a large number of credentials.

Test 2 – *Role and attribute increase*

Our next experiment is conducted to examine how well the authorization module scales when the roles and required attributes increase in *ROA* policies. Figure 9(c) shows the result when we increase the number of roles and attributes ranging from 1 to 100. To avoid confusions, each attribute is supported only by one credential in this experiment. In real collaborative environments, an originator does not likely define 100 roles, requiring 100 attributes to each role for the purpose of authorization. However, it is important to understand our system's behavior under extreme workload. In the worst case scenario, it takes less than 6 seconds for RAMARS Engine to go through all evaluations and derive an authorization decision. It is obvious in the graph that the policy loading and pre-processing time are major causes. We observe that the size of a single role assignment policy (*RAPS*) exceeds 3.5M bytes when 100 roles and relevant attributes are involved. As discussed in [20], the DOM XML parser is not efficient in handling large XML files. This could explain the high cost of policy loading and parsing. Better performance can be achieved by adopting a more efficient XML parser. We leave this issue as our future work since the development of efficient XML parser is beyond the scope of this paper.

As another effort in improving the performance of policy evaluation, we investigated bitmap-based indexing technologies [53] for the process of role assignment

evaluation. In bitmap indexing, data is abstracted as bit arrays, and queries are answered in a very efficient way by performing bitwise logical operations. We could further improve the performance to quickly locate the matched policies with the help of bitmap indexing and bitwise matching operations.

Consider *RAPS* as an example, *RAPS* specifies a set of role assignment policies, and each defines the required attributes for a particular role. The role assignment evaluation takes a set of a user's trusted attributes as an input and tries to derive the roles that could be assigned based on the role assignment policy. The performance of XACML PDP can be affected by scanning irrelevant policies that define the role with attributes other than the ones the user possesses. For example, if a policy defines role r_1 requires attributes a_1, a_2 and a_3, while a user has attributes a_4 and a_5, then the policy is definitely irrelevant to the evaluation. A bitwise match could be applied to expedite the matching process and only the relevant policies are fed into XACML PDP for policy evaluation. Suppose there are n attributes involved in the system (*ATTR* $= \{a_1, a_2, \ldots, a_n\}$). We follow the steps as illustrated below:

(1) Encoding a user's attributes into bitmap index: an n-bit bit array is introduced where the user's attribute is encoded as "1" at its position and "0" otherwise. As an example shown in Fig. 8(b), when there are 5 attributes involved in the system ($n = 5$), the user's attributes $\{a_1, a_2, a_3, a_4\}$ is encoded as a 5-bit bit array (11110).
(2) Encoding *RAPS* policies into bitmap index: for each policy element defining a role with a certain set of required attributes, an n-bit bit array is introduced. "1" is set to the position of each required attribute, and "0" otherwise. For example, when policy $P1$ defines r_1, requiring attributes $\{a_1, a_2, a_3\}$, it is encoded as $r_1 -$ (11100).
(3) Bitwise *OR* to find relevant policies: when a user's attributes cover all required attributes in a policy, the policy is a relevant policy that PDP needs to evaluate. In bitmap matching, a bitwise *OR* operation is performed between the user's attribute bit array and each policy bit array. If the derived bit array is equal to the user's bit array, the associated policy is a relevant one and should be promoted to PDP for policy evaluation. As shown in Fig. 8(b), after the bitwise match, only $P1$ and $P3$ are relevant ones for policy evaluation.

Theoretically, bitwise matching is more efficient than normal element-level matching in XML. Bitmap indexing could pre-parse the policies and quickly filter out relevant policies for PDP to evaluate. The more redundant policies are removed, the better performance PDP could achieve. We use t_1 to indicate the PDP evaluation time. However, the application of bitmap indexing also introduces extra burden in indexing policies and conducting bitwise matching operations. We use t_2 to indicate this overhead. In our experiments, we analyze the possible performance improvement that could be achieved by bitmap policy indexing. With 100 roles and 100 attributes involved in the system, there are 100 records of role assignment policies.

We adjust our original testing settings by varying the required attributes for each role and the user's attributes, so that we could control the number of relevant policies being promoted through bitmap indexing from 1 (the best case when bitmap indexing effectively removes 99 irrelevant policies) to 100 (the worst case when bitmap indexing finds all 100 policies are relevant ones). We measure and compare the time consumption of PDP evaluation alone (t_1) to the total time including the indexing process ($t_1 + t_2$). As shown in Fig. 9(d), the bitmap indexing has more performance advantage when there are more irrelevant policies in the policy pool. Along with the increase of relevant policies in the pool, the advantage of bitmap indexing is reduced by the cost of dynamic indexing and matching operations at runtime. However, we clearly observed that the robustness of bitmap indexing and its potential to possibly improve our policy evaluation process. Meanwhile, we also acknowledge that the bitmap indexing has limited expressiveness, and the bitwise comparison cannot replace the richness of XACML evaluation functionalities. Therefore, the application of bitmap indexing in our system is restrict to assist PDP to locate the relevant policies, while PDP is still responsible to evaluate the policies following the XACML evaluation procedures. In addition, some enhanced PDP engines [34] can be applied together with the bitmap indexing to further improve the performance of policy evaluation.

Test 3 – *RAMARS overhead to scientific P2P file sharing*

Our final experiment is conducted to measure the overhead of RAMARS authorization to the P2P based scientific file sharing infrastructure. The major overhead lies both at the requesting and the responding sides. On the requesting peer side, the requester sends supportive credentials along with the file request. On the responding peer side, the agent extracts the credentials from the file request, retrieves and evaluates ROA policies through RAMARS Engine.

According to a typical scientific collaboration of DZero Experiment (DØ) [19], 300 DØ users submitted 15,000 requests involving 2–4 TB data transfer per day with an average of 130M bytes data transfer per query. We choose a sample data file of size 121,781 kB to be transferred between two ShareEnabler agents for our experiment. We put one more monitor at the requesting peer side to measure the time between sending out the file request and finishing the file transfer. The base time is measured by turning off credential loading at the requester side and RAMARS Engine evaluation at the responder side. Both ShareEnabler agents are running within the same network domain. The requesting agent ran on a Pentium IV with 512 MB RAM running at 2.52 GHz with Windows XP. Table 1 shows the results of our experiment when we adjust the number of attributes, credentials and roles involved in the system. With the extreme complexity of evaluation, RAMARS introduces less than 5% overhead to the P2P file sharing infrastructure, which we believe is promising outcome with respect to the performance of ShareEnabler.

Table 1
RAMARS overhead analysis

Testcase (attr, cred, role)	Base	(10, 10, 10)	(20, 20, 20)	(30, 30, 30)	(40, 40, 40)	(80, 80, 80)	(100, 100, 100)
Total time (s)	487.5	488.0	489.0	490.8	492.8	499.8	508.3
Overhead (%)	0.00	0.10	0.30	0.68	1.07	2.53	4.27

7. Related work

There are a number of approaches being published in recent years to merge Trust Management (TM) approaches with RBAC. Li and Mitchell [29] and Shin and Ahn [45] consider TM as a form of distributed access control managing role-related credentials 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. In our approach, "role" is utilized by an originator to scope a virtual control domain for information dissemination, yet the trust decision making relies on more general-purpose credentials (e.g., affiliations, qualifications) for profiling and extending trust to unknown users within the domain. In [17], the authors attempt to extend conventional RBAC by introducing the notion of trust. In their scheme, users are first mapped to different trust levels, and the trust levels are then mapped to roles. As the trust level is closely associated with the role assignment, 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. In our approach, we aimed at an integrated and applicable framework for existing trust evaluation techniques to better serve for the access control in ad-hoc collaboration.

Trust-based approach has also been employed as an access control mechanism for social networks. In [7], the authors adopt a multi-level security approach using trust levels to classify both users and resources. A more sophisticated model has been proposed in [15]. The authors propose a semi-distributed discretionary access control model for controlled sharing of information in online social networks. In the access control model, users are authorized based on the type of relationship, depth, and trust level of existing relationships between nodes in the network. The authors also propose using certificates for granting relationships authenticity, and the client-side enforcement of access control according to a rule-based approach, where a user requesting to access an object must demonstrate that he has the rights of doing that. Compared to their approach, the authorization in our framework is not based on the requester's trust value. Instead, we apply a more comprehensive attribute-based access control for information sharing in ad-hoc collaboration, taking dissemination control as an important requirement in our framework. In addition, our trust model is based on delegation of attribute authority, therefore a different trust evaluation method is used to determine the trustworthiness of user attributes.

The trust-negotiation based approach [12,50,54] focuses on the bidirectional trust establishment between strangers through the iterative exchange of digital credentials. In our approach, we target a different goal to establish one-directional trust relationship from an originator extending to the collaborating users through delegations and trust evaluation. However, we see trust negotiation as a complimentary means and can be integrated into our framework for collaborating users to dynamically discover the originator's authorization requirements, and find out which credentials are necessary to satisfy the request. Shibboleth [14] is designed aiming at providing cross-domain single sign-on and attribute-based authorization. Shibboleth leverages the identity federation between educational organizations and establishes standardized attribute namespaces so that a user can authenticate at his own campus and access to remote resources where his attributes are passed to the resource providers for authorization. However, by focusing on the message protocols and attribute disclosures, very limited attribute-based authorization functionalities can be achieved by Shibboleth. Nevertheless, with its identity federation technology advances and established application base, Shibboleth can contribute to our framework by serving as the credential service to issue users' attribute assertions/credentials for RAMARS to evaluate.

A number of authorization systems have been developed to provide access control in distributed environments. There exists an approach to represent decentralize authorization using a language called SecPal [8]. Even though this language helps specify access control policies, it is less practical to support web-based collaboration systems. Instead, we leverage the features of XML to specify policies in our approach. The Akenti [5,47] system enforces access control on resources based on policies expressed by multiple distributed stakeholders. Akenti makes an extensive use of X.509 public key certificates as the authorization token for encoding both user attributes and usage conditions. PERMIS [16], on the other hand, leverages the role-based access control utilizing X.509 attribute certificates. However, it was proposed as authorization systems for localized resources, the originator could not remain control over the resource outside her administrative domain [23]. And the compatibility of Akenti and PERMIS with existing collaborative applications has never been extensively explored. The Community Authorization Service (CAS) [41] and Virtual Organization Membership Service (VOMS) [6] are two major authorization frameworks in Grid communities. Both CAS and VOMS systems establish a centralized service to issue policy assertions to a user within a Grid community, and the user presents the assertion to obtain certain access rights. The functionalities of CAS and VOMS heavily rely on centralized servers and a pre-established community administrator as the trust base, which prohibits them from supporting control-independent requirements in ad-hoc collaboration. In addition, [32] proposes a customized identity-based key agreement protocol to support the grid security infrastructure including a delegation protocol. However, the proposed protocol does not support characteristics involved in ad-hoc collaboration since it would create complex chain of proofs in handling and mapping identity attributes in such a dynamic collaboration. SciShare [10] and

LionShare [33] are among the few P2P based sharing systems that provide access control mechanisms. Both of the systems suit for ad-hoc collaboration in a sense that they achieve fully distributed membership and access control on every participant peer agent. However, the access control mechanisms in these two systems remain as a primitive ACL-based discretionary access control approach. In terms of trust evaluation, EigenTrust [27] and PeerTrust [52] develop a distributed trust mechanism for peer agents in P2P network to quantify and compare the trustworthiness of other peers based on their past interaction histories and reputations. We view ad-hoc collaboration as a community-based environment where limited trust relationships already exist among the collaborating organizations. Therefore, we adopt a semicentralized approach to controlling and computing the trust value based on the trusted delegation relationships between collaborating organizations, while leaving the issue of reputation out of our discussion.

8. Concluding remarks

In this paper, we proposed a comprehensive and concrete RAMARS framework to address trust management as well as access control requirements for secure information sharing in ad-hoc collaboration. RAMARS advocates the originator control, dissemination and delegation control by seamlessly leveraging the role-based approach with dynamic user abstractions and trust evaluation. Our framework is a policy-driven framework and the ShareEnabler system serves as the client-side reference monitor to enable the platform-to-platform policy enforcement and propagation in distributed collaboration environments.

Our future works are geared towards several directions. While the policy specification in supporting RAMARS model has been discussed in this paper, and the XACML-based policies provide great flexibility and expressiveness, the safety analysis of RAMARS has not been fully addressed. In particular, the dynamic property of an ad-hoc collaboration environment and distribution of trust authority makes it difficult for an originator to foresee any possibility of permission leakage at a certain system state. Therefore, a natural security concern is whether an originator still has some guarantees about whether her resource is shared within defined collaborative sharing domain. Studying the safety and decidability properties of the policy scheme [30], as well as the conformance checking of XACML policies [21], is going to give higher assurance for RAMARS framework. In addition, as addressed in our motivation example, sharing of medical records is restricted by many privacy protection regulations. The need-to-know principle must be strictly enforced for each responsible party to obtain only the necessary information to carry out its task. In particular, a user's medical profile is a complex composition of the patient demographics, medical history, examination and progress reports, medication and immunization status, laboratory test results, radiology images, billing records and so on. Assuring the authorized and selective sharing of such critical medical records

among several parties with different duties and objectives is indeed a huge challenge. Fine-grained control should be applied to support fully and partially sharing of such composite resource.

Acknowledgments

This work was partially supported by the grants from National Science Foundation and Department of Energy Early Career Principal Investigator Award.

Appendix

```
RolePolicySet = {RolePolicy} ;
RolePolicy    = [id] RoleName ("N"|"C") CapPolicyId  ;
id            = ? arbitrary string ? ;
RoleName      = ? arbitrary string ? ;
CapPolicyId   = ? arbitrary string ? ;

CapPolicySet           = CapPolicy, {CapPolicy} ;
CapPolicy              = CapPolicyId, (("N", (Operation, {Operation})) |
                         {"C", CapPolicyId_MappedRole)}, {CapPolicyId_JuniorRole} ;
CapPolicyId            = ? arbitrary string ? ;
Operation              = ? arbitrary access operation ? ;
CapPolicyId_MappedRole = CapPolicyId ;
CapPolicyId_JuniorRole = CapPolicyid ;

RoleAssignmentPolicySet = RAPolicy, {RAPolicy} ;
RAPolicy                = RoleName, (CombinationOP, (ReqAttrGroup, {ReqAttrGroup})),
                            {CombinationOP, (ReqAttrGroup, {ReqAttrGroup})} ;
ReqAttrGroup            = CombinationOP, (Aname, ComparisonOP,  TargetValue),
                            {Aname, ComparisonOP,  TargetValue} ;
CombinationOP           = "AND" | "OR" | "NOT" ;
ComparisonOP            = "="|"!="|">"|">="|"<"|"<=" ;
RoleName                = ? arbitrary string ? ;
Aname                   = ? arbitrary string ? ;
TargetValue             = ? arbitrary string ? ;

TAP_TLPolicySet = TAP_TLPolicy, {TAP_TLPolicy} ;
TAP_TLPolicy    = Attribute, {Attribute}, (Certifier, TrustValue), {Certifier, TrustValue} ;
Attribute       = Aname, [Avalue] ;
Certifier       = ? X.500 RFC2253 DN ? ;
TrustValue      = ? arbitrary real number between 0 and 1 ? ;

TAP_TDPolicySet = TAP_TDPolicy, {TAP_TDPolicy} ;
TAP_TDPolicy    = Attribute, {Attribute}, TrustValue, ComparisonOP, Threshold ;
Threshold       = ? arbitrary real number between 0 and 1 ? ;

RootMetaPolicy     = {(Resource, {Resource}), (ROAPolicyLocation, {ROAPolicyLocation})),
                       {(Resource, {Resource}), (ROAPolicyLocation, {ROAPolicyLocation})} ;
Resource           = (OriginatorId, {OriginatorId}), ResourceId ;
OriginatorId       = ? X.500 RFC2253 DN ? ;
ResourceId         = ? RFC2396 URI ? ;
ROAPolicyLocation  = ? RFC2396 URI ? ;

CREDPolicy     = Holder, Certifier, Attribute, {Attribute}, [(CombinationOP, Context, {Context})] ;
Context        = ValidityPeriod | Delegation ;
ValidityPeriod = Start, End ;
Delegation     = ? simple Boolean ? ;
Holder         = ? X.500 RFC2253 DN ? ;
Start          = ? arbitrary real number indicating date ? ;
End            = ? arbitrary real number indicating date ? ;
```

Fig. 10. Policy specification.

References

[1] ISO/IEC 14977, Information technology – syntactic metalanguage – extended BNF, 1996.

[2] A. Abdul-Rahman, The PGP trust model, *J. Electron. Commerce* **10**(3) (1997), 27–31.

[3] D.A. Agarwal and K. Berket, Supporting dynamic ad-hoc collaboration capabilities, *CoRR*, cs.OH/0307037, 2003.

[4] D.A. Agarwal, O. Chevassut, M.R. Thompson and G. Tsudik, An integrated solution for secure group communication in wide-area networks, in: *Proceedings of 6th IEEE Symposium on Computers and Communications*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2001, pp. 22–28.

[5] Akenti, Akenti distributed access control, http://dsd.lbl.gov/Akenti, 2009.

[6] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Á. Frohner, A. Gianoli, K. Lörentey and F. Spataro, VOMS, an authorization system for virtual organizations, in: *European Across Grids Conference*, Springer-Verlag, Berlin/Heidelberg, Germany, 2003, pp. 33–40.

[7] B. Ali, W. Villegas and M. Maheswaran, A trust based approach for protecting user data in social networks, in: *Proceedings of the Conference of the Center for Advanced Studies on Collaborative Research*, ACM Press, New York, NY, USA, 2007, pp. 288–293.

[8] M.Y. Becker, C. Fournet and A.D. Gordon, Secpal: design and semantics of a decentralized authorization language, *J. Comput. Secur.* **18** (2010), 619–665.

[9] K. Berket, D.A. Agarwal and O. Chevassut, A practical approach to the InterGroup protocols, *Future Gen. Comp. Syst.* **18**(5) (2002), 709–719.

[10] K. Berket, A. Essiari and A. Muratas, PKI-based security for peer-to-peer information sharing, in: *Proceedings of 4th IEEE International Conference on Peer-to-Peer Computing*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2004, pp. 45–52.

[11] T. Berners-Lee, R. Fielding and L. Masinter, Uniform resource identifiers (URI): generic syntax, RFC 2396, 1998.

[12] E. Bertino, E. Ferrari and A.C. Squicciarini, Trust-X: a peer-to-peer framework for trust establishment, *IEEE Trans. Knowl. Data Eng.* **16**(7) (2004), 827–842.

[13] M. Blaze, J. Feigenbaum, J. Ioannidis and A.D. Keromytis, The KeyNote trust-management system, RFC2704, 1999.

[14] S. Cantor (ed.), Shibboleth architecture, protocols and profiles, available at: http://shibboleth.internet2.edu/docs/internet2-mace-shibboleth-arch-protocols-latest.pdf, 2005.

[15] B. Carminati, E. Ferrari and A. Perego, Enforcing access control in web-based social networks, *Trans. Inform. Syst. Secur.* **13**(1) (2009), Article No. 6.

[16] D.W. Chadwick and A. Otenko, The PERMIS X.509 role based privilege management infrastructure, in: *Proceedings of 7th ACM Symposium on Access Control Models and Technologies*, ACM Press, New York, NY, USA, 2002, pp. 135–140.

[17] S. Chakraborty and I. Ray, TrustBAC: integrating trust relationships into the RBAC model for access control in open systems, in: *Proceedings of ACM Symposium on Access Control Models and Technologies*, ACM Press, New York, NY, USA, 2006, pp. 49–58.

[18] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos and R.L. Rivest, Certificate chain discovery in SPKI/SDSI, *J. Comput. Secur.* **9**(4) (2001), 285–322.

[19] I.T. Foster and A. Iamnitchi, On death, taxes, and the convergence of peer-to-peer and grid computing, in: *Proceedings of 2nd International Workshop on Peer-to-Peer Systems*, Springer-Verlag, Berlin/Heidelberg, Germany, 2003, pp. 118–128.

[20] S. Franklin, XML parsers: DOM and SAX put to the test, available at: http://www.devx.com/xml/Article/16922/0/page/1, 2001.

[21] V.C. Hu, E. Martin, J. Hwang and T. Xie, Conformance checking of access control policies specified in XACML, in: *Proceedings of 1st IEEE International Workshop on Security in Software Engineering*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2007, pp. 275–280.

[22] ITU-T Rec. X.509 ISO/IEC 9594-8, The directory: public-key and attribute certificate frameworks, 2001.

[23] W. Jie, J. Arshad, R. Sinnott, P. Townend and Z. Lei, A review of grid authentication and authorization technologies and support for federated access control, *ACM Comput. Surv.* **43**(12) (2011), 1–26.

[24] 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*, ACM Press, New York, NY, USA, 2006, pp. 200–209.

[25] J. Jin, G.-J. Ahn, M. Shehab and H. Hu, Towards trust-aware access management for ad-hoc collaborations, in: *Proceedings of 3rd IEEE International Conference on Collaborative Computing*, New York, NY, USA, 2007.

[26] J. Jin, G.-J. Ahn and M. Singhal, ShareEnabler: policy-driven access management for ad-hoc collaborative sharing, in: *Proceedings of 2nd International Workshop on Pervasive Information Management*, Springer-Verlag, Berlin/Heidelberg, Germany, 2006, pp. 724–740.

[27] S.D. Kamvar, M.T. Schlosser and H. Garcia-Molina, The eigentrust algorithm for reputation management in P2P networks, in: *Proceedings of the 12th International Conference on World Wide Web*, W3C, MA, USA, 2003, pp. 640–651.

[28] G. Karjoth, M. Schunter and M. Waidner, Platform for enterprise privacy practices: privacy-enabled management of customer data, in: *Proceedings of 2nd International Workshop on Privacy Enhancing Technologies*, Springer-Verlag, Berlin/Heidelberg, Germany, 2002, pp. 69–84.

[29] N. Li and J.C. Mitchell, RT: a role-based trust-management framework, in: *Proceedings of 3rd DARPA Information Survivability Conference and Exposition*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2003, pp. 201–212.

[30] N. Li, J.C. Mitchell and W.H. Winsborough, Beyond proof-of-compliance: security analysis in trust management, *J. ACM* **52**(3) (2005), 474–514.

[31] N. Li, W.H. Winsborough and J.C. Mitchell, Distributed credential chain discovery in trust management, *J. Comput. Secur.* **11**(1) (2003), 35–86.

[32] H. Lim and K. Paterson, Identity-based cryptography for grid security, *Int. J. Inform. Secur.* **10** (2011), 15–32.

[33] LionShare project, http://lionshare.its.psu.edu/main/.

[34] A.X. Liu, F. Chen, J. Hwang and T. Xie, Xengine: a fast and scalable XACML policy evaluation engine, in: *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems*, ACM Press, New York, NY, USA, 2008, pp. 265–276.

[35] M. Mohri, Semiring frameworks and algorithms for shortest-distance problems, *J. Automata Languages Combinat.* **7**(3) (2002), 321–350.

[36] NIST, All simple paths, http://www.itl.nist.gov/div897/sqg/dads/HTML/allSimplePaths.html.

[37] NIST, Single-source shortest-path problem, http://www.itl.nist.gov/div897/sqg/dads/HTML/singleSourceShortestPath.html.

[38] OASIS, Core and hierarchical role based access control (RBAC) profile of XACML v2.0, available at: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf, 2005.

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

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

[41] L. Pearlman, V. Welch, I.T. Foster, C. Kesselman and S. Tuecke, A community authorization service for group collaboration, in: *Proceedings of 3rd International Workshop on Policies for Distributed Systems and Networks*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2002, pp. 50–59.

[42] R.L. Rivest and B. Lampson, SDSI: a simple distributed security infrastructure, available at: http://research.microsoft.com/lampson/59-SDSI/WebPage.html, 1996.

[43] R.S. Sandhu, E.J. Coyne, H.L. Feinstein and C.E. Youman, Role-based access control models, *IEEE Computer* **29**(2) (1996), 38–47.

[44] H. Shen and P. Dewan, Access control for collaborative environments, in: *Proceedings of ACM Conference on Computer-Supported Cooperative Work*, ACM Press, New York, NY, USA, 1992, pp. 51–58.

[45] D. Shin and G.-J. Ahn, Role-based privilege and trust management, *Comput. Syst. Sci. Eng.* **20**(6) (2005), 401–410.
[46] Swedish Institute of Computer Science, XACML 3.0 administrative policy support (beta version), available at: http://www.sics.se/spot/xacml_3_0.html, 2006.
[47] M.R. Thompson, A. Essiari and S. Mudumbai, Certificate-based authorization policy in a PKI environment, *ACM Trans. Inf. Syst. Secur.* **6**(4) (2003), 566–588.
[48] M. Wahl, S. Kille and T. Howes, Lightweight directory access protocol (v3): UTF-8 string representation of distinguished names, RFC 2253, 1997.
[49] Wikipedia, Branching factor, available at: http://en.wikipedia.org/wiki/Branching_factor.
[50] W. Winsborough and N. Li, Towards practical automated trust negotiation, in: *Proceedings of IEEE Workshop on Policies for Distributed Systems and Networks*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2002, pp. 92–103.
[51] M. Winslett, C.C. Zhang and P.A. Bonatti, Peeraccess: a logic for distributed authorization, in: *Proceedings of the 12th ACM Conference on Computer and Communications Security*, ACM Press, New York, NY, USA, 2005, pp. 168–179.
[52] L. Xiong and L. Liu, PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities, *IEEE Trans. Knowl. Data Eng.* **16** (2004), 843–857.
[53] J.P. Yoon, Presto authorization: a bitmap indexing scheme for high-speed access control to XML documents, *IEEE Trans. Knowl. Data Eng.* **18**(7) (2006), 971–987.
[54] T. Yu and M. Winslett, A unified scheme for resource protection in automatic trust negotiation, in: *Proceedings of the IEEE Symposium on Securiy and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, USA, 2003, pp. 110–122.
[55] L. Zhang, G.-J. Ahn and B.-T. Chu, A rule-based framework for role-based delegation and revocation, *ACM Trans. Inf. Syst. Secur.* **6**(3) (2003), 404–441.