ASSURED INFORMATION SHARING FOR AD-HOC COLLABORATION

by

Jing Jin

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Information Technology

Charlotte

2009

Approved by:

Dr. Gail-Joon Ahn

Dr. Mohamed Shehab

Dr. Cem Saydam

Dr. John A. Gretes

ABSTRACT

JING JIN. Assured information sharing for ad-hoc collaboration.
(Under the direction of DR. GAIL-JOON AHN)

Collaborative information sharing tends to be highly dynamic and often ad hoc among organizations. The dynamic natures and sharing patterns in ad-hoc collaboration impose a need for a comprehensive and flexible approach to reflecting and coping with the unique access control requirements associated with the environment.

This dissertation outlines a Role-based Access Management for Ad-hoc Resource Sharing framework (RAMARS) to enable secure and selective information sharing in the heterogeneous ad-hoc collaborative environment. Our framework incorporates a role-based approach to addressing originator control, delegation and dissemination control. A special trust-aware feature is incorporated to deal with dynamic user and trust management, and a novel resource modeling scheme is proposed to support fine-grained selective sharing of composite data. As a policy-driven approach, we formally specify the necessary policy components in our framework and develop access control policies using standardized eXtensible Access Control Markup Language (XACML). The feasibility of our approach is evaluated in two emerging collaborative information sharing infrastructures: peer-to-peer networking (P2P) and Grid computing. As a potential application domain, RAMARS framework is further extended and adopted in secure healthcare services, with a unified patient-centric access control scheme being proposed to enable selective and authorized sharing of Electronic Health Records (EHRs), accommodating various privacy protection requirements at different levels of granularity.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor Professor Gail-Joon Ahn. His expert guidance and support have made this work possible, sparked my interest in access control, and were the foundation of a great graduate research experience.

Many thanks to the numerous individuals who worked with me on collaborative papers, and on other topics related to my research over the last few years, including Professor Mohamed Shehab from UNC Charlotte, Dr. Xinwen Zhang from Samsung Information Systems America, Dr. Michael J. Covington from Intel Corporation, and graduate students Hongxin Hu and Wenjuan Xu. Their feedback and criticism enriched my work and helped me to align it with other research projects.

I also wish to express my deep appreciation to my PhD committee members, Professor Gail-Joon Ahn, Professor Mohamed Shehab, Professor Cem Saydam and Professor John A. Gretes. The assistance and guidance they provided in the preparation of this manuscript have been invaluable.

Furthermore, I want to acknowledge the multi-year funding I received for this work via research grants from National Science Foundation (NSF-IIS-0242393) and Department of Energy Early Career Principal Investigator Award (DE-FG02-03ER25565) to Professor Gail-Joon Ahn.

Finally, and most importantly, I would like to thank my husband Yunjun. His support, encouragement, patience and unconditional love enabled me to surpass hardships and complete this work. I also thank my parents Lingen and Wenying. They were always supporting me and encouraging me with their best wishes.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

The rise of Internet has led collaborators to face dramatic changes in managing and sharing their resources. Subsequently, it has extremely influenced to the traditional information sharing fashion. Collaborative information sharing has increasingly turned outward to connect distributed participants across enterprise and institutions. By removing the barriers of the time and geographical distance from the conventional collaborations, people are able to work together regardless of their locations. And new terms such as *virtual organization*, *virtual laboratory*, and *collaboratorium* have been introduced. Such a new paradigm of collaboration and organization settings has turned the information sharing into a widely spread and highly dynamic network-based environment, while formally accessing the resources in a secure manner poses a difficult challenge. This dissertation addresses the issues of advocating selective and secure information sharing for web-based collaborative environments.

1.1 Ad-hoc Collaboration

Under many circumstances, the collaboration relationship among distributed parties is established based on spontaneous interactions and use patterns in an ad-hoc fashion. For example, groups of universities, laboratories, and industrial companies may collaborate and mutually share research results for treatment of a particular human disease; different educational agencies collaboratively implement, disseminate, and institutionalize effective practices for supporting and promoting people from underrepresented groups in the field of Computer Science; and federal, state and local government agencies share information to coordinate the efforts of preventing terrorist attacks. As these examples indicate, a variety of distributed parties may need to collaborate with entities that they do not necessarily

trust. And the establishment of collaboration relationship is highly dynamic and may vary tremendously in purpose, scope, size, duration, and the number of participants. In this dissertation, we informally denote such a new type of collaboration environment as *ad-hoc collaboration* [105, 7, 67, 66]. Our definition of ad-hoc collaboration is as follows:

> *Based on common interests, distributed individual participants who belong to many different organizations spontaneously establish or join collaborations, and dynamically perform a variety of activities such as communication, information sharing, cooperation, problem solving, and negotiation.*

Among the various activities performed in ad-hoc collaboration, in this dissertation we focus on the digital information sharing. Compared to well-structured and managed internal collaborations, the formulation of ad-hoc collaboration is more transient and there is no pre-established global consensus of trustworthiness among all distributed participating parties.

1.2 Emerging Collaborative Sharing Infrastructures

Traditionally, collaborative information sharing heavily relies on client-server based approaches or email systems. By recognizing the inherent deficiencies such as a central point of failure and scalability issue, several distributed computing alternatives have been proposed to support collaborative sharing of resources, including Grid computing [48, 19] and Peer-to-Peer (P2P) networking [88].

Grid computing [48, 19] is a distributed computing infrastructure for advanced science and engineering applications to share remote resources, federate datasets, and pool computers for large-scale simulations and data analysis. Researchers of many disciplines rely on the established infrastructure to collaboratively compute and solve data intensive problems that are too complex for the resources of a single institution. In Grid computing, the collaboration is organized in a form of *virtual organization* (VO) including dynamic collections of geographically distributed individuals, institutions and resources. With open standards

and communication protocols, the Grid computing infrastructure allows VO members to uniformly share resources regardless of their locations and internal architectures.

While Grid computing suits for highly structured scientific collaborations with established infrastructures, P2P networking works well on heterogeneous network environments and promises to be more flexible and reliable for smaller ad-hoc collaborative interactions [17, 45]. A pure P2P network does not have the notion of clients or servers but only equal peer nodes that simultaneously function as both "clients" and "servers" to the other nodes on the network. Especially, with its decentralized structure and load balancing feature, P2P-based file sharing system offers better scalability and robustness connecting millions of simultaneous peer nodes. As demonstrated in the newly proposed SciShare system [17, 18], P2P networking technology also has great potentials to support information sharing in collaborative environments.

1.3 Access Control Challenges

Nevertheless, given the diverse contexts of collaborative sharing and heterogenous supportive infrastructure settings, achieving effective access control is a critical requirement. The sharing of sensitive information in collaboration is necessarily to be highly controlled by defining what and how is to be shared, who is allowed to share and under which condition. This becomes extremely difficult in an open and dynamic environment when a large number of collaborating users may leave or join the collaboration at any time.

In ad-hoc collaboration, users that have no pre-existing relationships may try to collaborate and share information. Therefore, the resource provider or authorization authority should be able to cope with a large number of strangers. In this case, it has made the traditional identity-based access control approaches generally ineffective, as these approaches rest on the principle that a user requesting access should be known a priori [33, 102, 44]. Instead, the critical issue in an open and dynamic collaborative environment is not "Who exactly is this requester," but "Do I trust this requester to share my resource?" The properties or attributes possessed by the requesting users (i.e., employment status, citizenship,

group membership and qualifications) are more relevant to characterizing users and determining whether or not they should be trusted to conduct sensitive interactions for information sharing. Usually, user properties or attributes are carried in a form of *credentials*, such as academic diplomas, certifications, security clearances, and identification documents. Meanwhile, the same attribute may be attested through a number of different credentials. For instance, a user's name may be testified in his driver's license by a local DMV office or in his passport by the U.S. government agency. However, when user attributes are utilized as a generalized access control mechanism, different credentials may not always be trusted to the same extent and thus fail to assert the attributes, resulting in the denial of access to the shared data. Therefore, dealing with such trust management concerns is extremely important in order to achieve effective access control in ad-hoc collaboration.

Another important aspect involved in collaborative sharing is to provide fine-grained selective access control for the shared resources. In particular, the resource being shared in a collaboration may be a composition of different types of sub-objects. Such resources need to be shared fully or partially depending on different collaboration purposes and degrees of sensitivity. A fine-grained access control mechanism is thus required to support a spectrum of protection granularity levels, where different access control policies are applied to different portions of the same resource. As a motivation example, a patient's electronic health records may include the patient's identification information, insurance information, the doctor's written notes for each office visit, lab results, X-ray images and so on. However, disclosing the complete patient's profile to collaborators without discretion can cause a great breach of the patient's privacy. Sharing of such complex resources must comply with legal and regulatory policies while simultaneously ensuring that access to sensitive information is limited only to those entities who have a legitimate need-to-know requirement. For instance, only the patient's disease symptoms and related medical histories in the profile need to be shared with other health care professionals for disease diagnosis and research purposes, while the sensitive personal identifiable information should be well protected.

Given these complexities, we need to address several challenging questions:

- What are the unique access control requirements in ad-hoc collaboration?
- How trust can be managed among strangers within the collaboration? And to what extent can trust management contribute to the access control solutions?
- How to promote fine-grained and selective information sharing among disparate collaboration parties?
- How can we build secure and interoperable systems to enable collaborative sharing in heterogeneous network environments?

These are critical questions to be answered to assure the secure information sharing in ad-hoc collaboration. Some approaches have been proposed – from formal access control models to security policies, mechanisms and systems – to address information sharing and general access control issues in dynamic environments [102, 10, 22, 38, 72, 65, 73, 5, 92, 11, 108, 27, 18, 76]. However, as ad-hoc collaboration is a newly emerged environment, our study clearly indicates that there is a need to design a comprehensive access management framework that is general and flexible enough to cope with the special access control as well as trust management requirements associated with the environment. In this dissertation work, we would make one step towards this direction.

1.4 Statement of the Hypothesis and Approaches

Therefore, this research hypothesizes that:

An effective access management framework is the key to enabling secure information sharing in ad-hoc collaboration.

To accomplish our goal in this dissertation work, we consider two fundamental security engineering questions to answer: "what" security needs to be enforced, and "how" the security is enforced [101]. In ad-hoc collaboration, the environment is highly dynamic and distributed, spanning organizational boundaries. Trying to answer these two questions in a

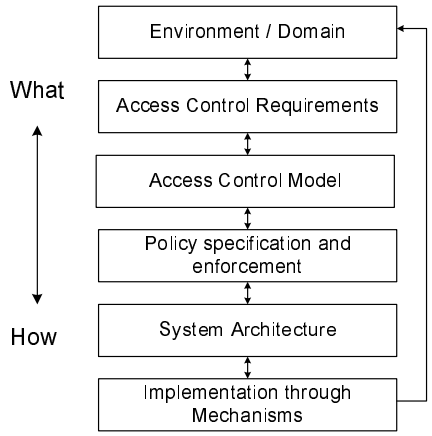


Figure 1.1: Multi-layered Research Method

single step is hardly viable in such a complex environment. As a systematic approach to analyzing and solving the problem, we first adopt an intuitive layered research process. As illustrated in Figure 1.1, we need to clearly understand the target environment (or problem domain) to identify access control requirements to be fulfilled in security solutions. These requirements, however, are often informal since they are derived from a particular business domain. From the security engineering domain, an access control model then should be proposed to formally articulate these access control requirements. And based on the identified formal access control model, we need to specify policies to realize the concepts and components conveyed in the model. While the access control model and policy framework focus on the "what" aspect, the system architecture and implementation mechanisms address the "how" aspect in terms of detailed system components and operational procedures to enforce the desired access control policies. Finally, the security system must be evaluated against the target business domain to examine how well the access control requirements have been accommodated. The main purpose of this approach is to guide our research to proceed in a systematic way while keeping clear as to which decisions are being made at each layer.

Following this research process, this dissertation work designs and develops an access management framework advocating selective and secure information sharing for ad-hoc

collaboration. In particular, we first scope and analyze the ad-hoc collaboration environment with a concrete scenario of public health surveillance collaboration. From the scenario, we articulate the generic access control requirements and trust management issues associated with the environment, and then we analyze the patterns of collaborative information sharing and dissemination, from which more detailed requirements for secure information sharing are derived. These requirements are reflected and addressed in our proposed Role-based Access Management for Ad-hoc Resource Sharing framework (RAMARS). In particular, RAMARS framework incorporates a role-based approach to addressing originator control, delegation and dissemination control, with a special trust-aware feature being incorporated to deal with dynamic user and trust management. In order to evaluate the feasibility of our approach, RAMARS framework has been applied in both P2P networking and Grid computing infrastructures to support secure and authorized information sharing in collaborations. In addition, we propose a comprehensive evaluation matrix to measure the performance of our systems. Finally, we adopt RAMARS framework in healthcare domain with an extended composite resource modeling and authorization scheme to enable selective sharing of patients' Electronic Health Records (EHRs) at different levels of granularity, accommodating various privacy protection requirements at different levels of granularity.

1.5 Summary of Contributions and Dissertation Organization

The contributions of our RAMARS framework are summarized as follows:

- We demonstrate a systematic research methodology to provide comprehensive and realistic security solutions to newly emerged environments.
- We articulate the unique access control requirements and trust management concerns of ad-hoc collaboration environments to address the issues of selective and secure resource sharing, minimizing the risks of unauthorized access.
- We propose a comprehensive and integrated solution for supporting secure and selective sharing in ad-hoc collaboration, incorporating ideas of role-based access control,

originator control, delegation, trust management and fine-grained access control on composite resources.

- We demonstrate a policy-driven approach to achieving security in distributed computing environments. A formal policy specification scheme is proposed and implemented based on eXtensible Access Control Markup Language (XACML).
- We propose a comprehensive and well-designed system architecture that highlights key design issues for distributed authorization systems.
- We provide first-hand experience in building practical authorization systems for both P2P networking and Grid computing communities with rigorous performance evaluation and system enhancement.
- We propose and demonstrate a possible approach to supporting privacy preserving medical information sharing for secure healthcare services.

The remainder of this dissertation is organized as follows. Chapter 2 discusses general security fundamentals and reviews existing authorization mechanisms for collaborative environments. In Chapter 3, we clearly define the scope of ad-hoc collaboration and analyze the unique access control requirements within the environment. Chapter 4 elaborates the detailed design and formalization of RAMARS framework. The policy specification and evaluation is discussed in Chapter 5. Concrete implementations of the RAMARS framework in the context of P2P networking and Grid computing infrastructures are described in Chapter 6. A case study of possible applications of RAMARS framework in healthcare domain with fine-grained authorization and selective sharing feature is discussed in Chapter 7. Finally Chapter 8 summarizes this dissertation and presents some directions for future work.

CHAPTER 2: BACKGROUND AND RELATED WORK

In order to fully leverage an effective access control solution for secure information sharing in ad-hoc collaboration, it is important to have thorough knowledge of related access control models, the available security management mechanisms and existing security systems for collaborative environments.

In an access control model, the entities that can perform actions in the system are called *subjects*, and the entities representing resources to which access may need to be controlled are called *objects*, and the ways in which a subject can access an object are called *access rights*. A sound understanding of existing access control models helps us analyze the strength and weakness of an access control model in regulating the subjects, objects and access rights in its perspective domain, which in return help us design a comprehensive access control model for the newly emerged ad-hoc collaboration environment.

The security management mechanisms elaborate the state-of-the-art technologies to realize various access control models as concrete access control policies, security mechanisms and security architectures in building real security management systems. These technologies are also utilized as part of our solutions to implement our proposed approach and build secure information sharing systems for ad-hoc collaboration.

The literature review of existing security systems for collaborative environments lead us a better understanding of the current state of security research in the field, so that we

could justify our research approach, compare and evaluate our security solutions.

2.1 Related Access Control Models

2.1.1 Traditional Access Control Models

Historically and traditionally, there are two fundamental types of access control models defined by the Department of Defense Trusted Computer Security Evaluation Criteria (TCSEC) for the protection of data from unauthorized access and disclosure [33]: mandatory access control (MAC) and discretionary access control (DAC).

MAC has been closely associated with multi-level secure (MLS) systems [41]. Information (object) is classified with a security label based on its sensitivity, which often ranges from *unclassified* to *classified*, *secret*, *top secret* and so forth. A user (subject) is associated with a certain security attribute, called clearance. The access control decision is then contingent on verifying the compatibility of the security labels of the information and the clearance properties of the requesting user. Based on elegant mathematics models, MAC models could achieve a high degree of robustness and assurance, and their access control policies strongly restrict the scope and direction of information dissemination within a closed environment [16, 32].

By contrast, DAC models [70, 103] are defined as "a means of restricting access to objects based on the identity of subjects and/or groups to which they belong" [33]. The access control list (ACL) [103] is a typical implementation of DAC model that maintains a list of permitted subjects with their permissions attached to an object. The main difference from MAC is that in DAC the owner of an object has discretionary authority over who else can access to the object. DAC mechanisms are often not used in a stringent need-to-know environment because they are widely considered lack of the required safety assurance [79].

During the last decade, Role-Based Access Control (RBAC) model has been widely accepted as an alternative approach to MAC and DAC for managing and enforcing security in large-scale and enterprise-wide systems [102, 44]. The central construct of RBAC is the role. Permissions are associated with roles, and users acquire permissions by be-

ing assigned to appropriate roles. Roles are always created based on the set of necessary permissions to execute a specific job function in an organization, and organizational users are then assigned according to their responsibilities and qualifications. Users can be easily reassigned from one role to another. And permissions can be easily granted to or revoked from roles as needed. With the abstraction between users and permissions through roles, RBAC could tremendously reduce the complexities of security management for system administrators. RBAC96 models and the later NIST standard RBAC model [102, 44] also include components of role hierarchies and constraints. Role hierarchies are a natural means for structuring roles to reflect an organization's lines of authority and responsibility. In the hierarchical structures, senior roles generally inherit the permissions assigned to junior roles, and this enables the role construct to be multi-layered, thereby further reducing the number of relations between users and permissions. Constraints are applied to establish higher level organizational policies. A well-known example of constraints is the separation of duty. For instance, the same user cannot be a member of both `purchasing manager role` and `accounts payable manager role`. The separation of duty constraint reduces possible frauds or errors by controlling membership and use of roles as well as permission assignment [102]. Figure 2.1 illustrates the basic elements and system functions in the RBAC96 models. With all these features, RBAC systems enable users to carry out a broad range of authorized operations and provide great flexibility and breadth of application. It has been demonstrated that both MAC and DAC access control policies can be configured using RBAC [89].

In traditional closed environments, access control policies rest on the principle that a user requesting access should be known a priori. As a result, it is assumed that a user's identity information should exist within policy-effective domain(s) where an access control decision can be made. MAC, DAC and RBAC all remain in line with the principle. They regulate subjects' access on the basis of their known security clearances, their identities or groups, and roles, respectively. We generalize such access control models as identity-based

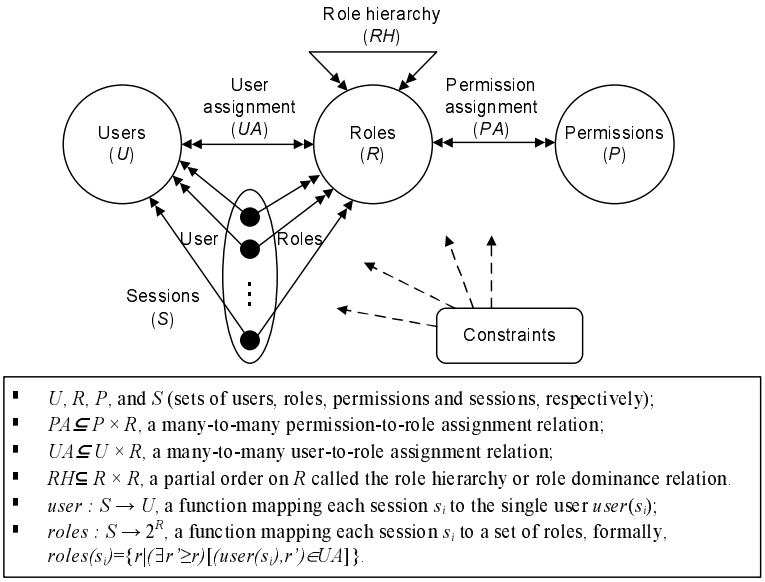


Figure 2.1: Basic Elements and Functions in RBAC96 Model

access control models. However, these conventional access control approaches are often inadequate to meet all the requirements that today's open network environments usually set out when the aforementioned principle does not hold any more [74, 109].

2.1.2 Delegation and Trust Management

Delegation has been recognized to leverage a means of propagating authorities and responsibilities in a distributed computing environment. Over years, many role-based delegation models [116, 10, 14] have been proposed to formulate and control the behaviors where organizational users themselves delegate role authorities to others to carry out some functions authorized on behalf of the former.

Trust management (TM) [23], as another generalized approach to utilizing delegation, has attracted growing attention for authorization in open and distributed network environments. Central to TM approaches is the capability delegation, which is incorporated into a credential chain from a resource owner to an access requester. An example usage of TM is shown in Figure 2.2. The TM approach generally deals with the following components for its proper functionality.

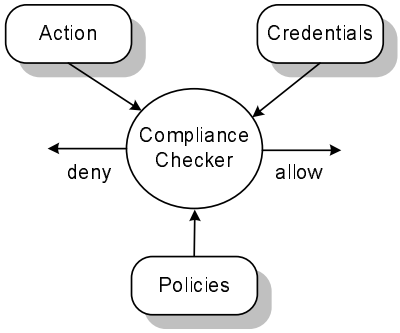


Figure 2.2: Trust Management

- Policies: they serve as the ultimate sources of authority (i.e., resource owner) in the local environment for controlling access.
- Credentials: they represent delegations of trust to prove the authorization of certain requested actions.
- Compliance Checker: this represents an algorithm or mechanism for checking if a set of credentials proves that a requested action complies with the policies.

Unlike traditional identity-based access control models, TM approaches essentially belong to the attribute-based access control that grants access to users based on the attributes possessed by the requester without assuming the users being known a priori. Hence, it provides an authorization framework for unknown entities, whereby it is possible to express, evaluate, and enforce decentralized access control policies and credentials in an open and distributed computing environment. KeyNote, SPKI/SDSI, Delegation Logic (DL), SD3 and RT are all examples of TM systems designed along the line [22, 38, 72, 65, 73].

Even though the notion of "trust management" has been widely utilized, the TM systems limit the scope of "trust" for the purpose of distributed access control, where the trust relationship between the requesting user and the local resource owner is treated as a boolean value to directly derive the authorization decision. A common assumption made in these systems is that the requesting user's credentials are unconditionally accepted. However, this may not be the case as trust is always considered as a subjective notion to the trusting

entity, and some entities may be trusted more than others with respect to performing an action [51, 34]. Therefore, the meaning of trust and the role of trust management in the context of collaborative sharing environments need to be carefully examined before any adoption of trust management approaches.

2.1.3 Models for Dissemination Control

Originator control (ORCON) is a special access control policy defined by a resource owner, so called *originator*, to control the dissemination of restricted resources [5, 79]. In contrast to MAC models where the authorization policy for information access and dissemination is uniformly driven by security labels of the resources, ORCON shares with DAC the notion that enables the resource owner to decide who should have access to it with a special originator control label. ORCON policies also require that resource recipients obtain the owner's permission to re-disseminate protected resources. The enforcement of ORCON policies relies on a centralized control authority to maintain the ORCON resource labellings, and a non-discretionary access control list to maintain the designated recipients of the resource. Such approach limits the ability to enforce ORCON policies outside of a closed system environment where a central control authority such as a reference monitor is not available [90].

The concept of Usage Control (UCON) is introduced in [90, 91] for controlling access and usage of digital information objects. In UCON, authorization decisions are not only checked and made before the access, but may be continuously checked during the access. Access rights may be revoked if some policies are not satisfied any more during the access when the context of subjects, objects or environmental conditions changes. The re-dissemination control addressed in ORCON is also one of the key concerns in UCON. By introducing licensing and ticketing mechanisms [90], UCON has the potential to support and enforce ORCON policies in more versatile and flexible ways for distributed environments. Most recently, the notion of dissemination control (DCON) has been introduced with a broader concept concerning with controlling information during the usage and dis-

semination activities [107].

The aforementioned models, although very relevant to controlling the digital information sharing and dissemination in collaborative environments, mostly stay at a conceptual and theoretical level. They lack of practical technologies or systems to support the concepts and functions illustrated in these models.

2.2 Security Management Mechanisms

2.2.1 Authentication and Authorization Infrastructures

The public key cryptography has been widely used for encryption and digital signatures to ensure data security and integrity during message exchanges between communicating parties. To verify the authenticity of an entity's identity or the signature on a signed message, one needs to have that entity's public key. In unsecured network environments such as Internet, it is essential to verify the authenticity of the entity itself and make sure the entity's keys are not compromised. A Public Key Infrastructure (PKI) [6] is thus introduced to leverage commonly trusted entities to securely associate identities with public keys. In a PKI, a subject's identity is bound to a public key via a certificate issued and signed by a commonly trusted Certification Authority (CA). A CA issues digitally signed certificates to certify that a specific identity to public key mapping is authentic. A widely used standard identity certificate is the X.509 Public Key Certificate (PKC) as specified in RFC3280 [58]. Figure 2.3 illustrates the structure of an X.509 PKC. As a subject's key may be compromised or the mapping of the key to identity may become invalid, a Certificate Revocation List (CRL) should be publicly made available to all relying parties that accept PKCs for authentication.

While a PKI is generally for authentications that verify the authenticity of an entity's identity, a Privilege Management Infrastructure (PMI) manages an entity's authorizations as the entity's attributes [64]. As illustrated in Table 2.1, there are many similar concepts in PKIs and PMIs [28]. While an X.509 PKC is used to maintain a strong binding between

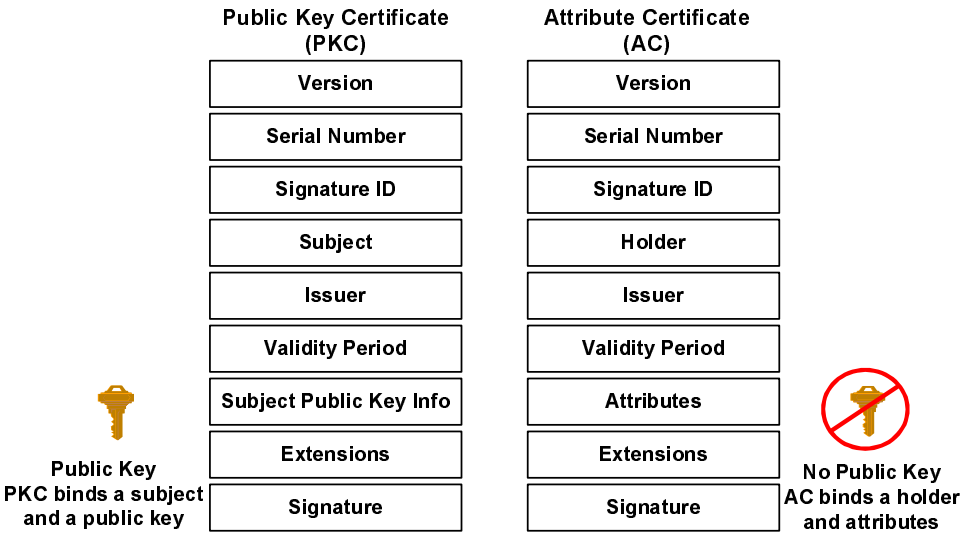


Figure 2.3: Public Key Certificate and Attribute Certificate

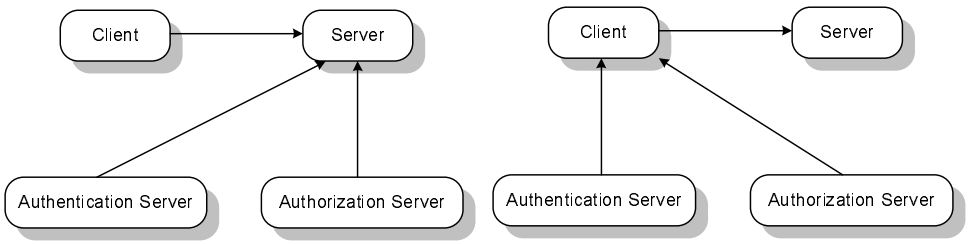
Table 2.1: A Comparison of PKIs with PMIs

Concept	**PKI Entity**	**PMI Entity**
Certificate	Public Key Certificate	Attribute Certificate
Certificate issuer	Certification Authority (CA)	Attribute Authority (AA)
Certificate user	Subject	Holder
Certificate binding	Subject's name to public key	Holder's name to privilege attribute(s)
Revocation	Certificate revocation list (CRL)	Attribute certificate revocation list (ACRL)
Root of trust	Root CA	Source of authority (SOA)

a user's identity and his/her public key, an X.509 Attribute Certificate (AC) [40] maintains a strong binding between a user's identity and arbitrary attributes (i.e., role membership information, policy statements, accounting information) for authorization purposes. The entity that signs an AC is called an Attribute Authority (AA). And the root of trust of the PMI is called the Source Of Authority (SOA). If a user needs to have authorization permissions revoked, an AA maintains an Attribute Certificate Revocation List (ACRL) available to all relying parties for authorization.

2.2.2 Security Enforcement Architecture Pull Vs. Push

In security engineering, the *pull model* and the *push model* designate two well-known approaches to exchanging security data between a requesting client and an enforcement server. The *pull model* is based on the request-response paradigm: the client sends an



(a) Pull Architecture (b) Push Architecture

Figure 2.4: Security Enforcement Architecture

access request to the server, then the server is responsible to obtain the client's authentication and authorization information in order to enforce the authorized access to the client. The *push model*, in contrast, requires the client to deliver his/her authentication and authorization credentials at the initial access request. Figure 2.4 illustrates these two system enforcement architectures.

One significant advantage of the *push* approach is that the enforcement server is provided with the information required to take an access decision immediately. The computational and communication overheads of making an access decision using a push model are negligible in principle. Especially, the push model has advantages in utilizing PKIs and PMIs when the information contained in the certificate is static and the validity of the certificate can be assumed. However, if the information contained in the certificate is liable to change or the certificate may be revoked, the *pull* approach is necessary for the enforcement server to check that no CRL or ACRL exists for a certificate each time an access request is made for authentication and authorization purposes. Clearly, this imposes considerable computational and communication overheads on the decision-making process at the server side, yet simplifies the overall interception by the client.

2.2.3 eXtensible Access Control Markup Language (XACML)

The eXtensible Access Control Markup Language (XACML) is an OASIS standard general purpose policy specification language as well as an access decision language [84, 85]. XACML consists of a syntax definition for a policy language and specifies semantic rules

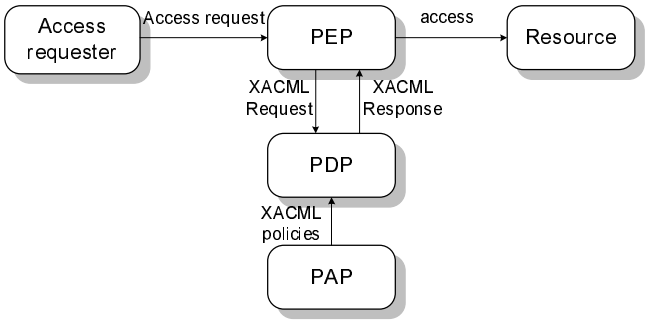


Figure 2.5: XACML Evaluation Flow

to process and evaluate XACML policies.

A typical setup for XACML policy evaluation is illustrated in Figure 2.5. An access request is sent to a Policy Enforcement Point (PEP) that protects the resource. The PEP forms an XACML *Request* based on the requester's attributes (*Subject*), the resource in question (*Resource*), the action (*Action*), and other information pertaining to the request (*Environment*). The PEP then sends this XACML *Request* to a Policy Decision Point (PDP), which evaluates the request against XACML policies that applies to the request. XACML policies are maintained and provided through a Policy Administration Point (PAP). An XACML *Response* is returned from PDP back to PEP with an authorization decision, and PEP is responsible to enforce the decision accordingly. An authorization decision included in an XACML *Response* pertains to the following four values:

- PERMIT: The requested action is allowed.
- DENY: The requested action is not allowed.
- NOT APPLICABLE: The policies available to PDP are not applicable to the request.
- INDETERMINATE: Some errors occurred during evaluation.

The root of XACML policies is a *Policy* or a *PolicySet*. A *PolicySet* is a container that can hold multiple *Policies* or *PolicySets*, as well as references to policies found in remote locations. A *Policy* represents a single access control policy, expressed through a set of *Rules*. Each *Rule* has a boolean function *Condition* that is associated with an *Effect* of

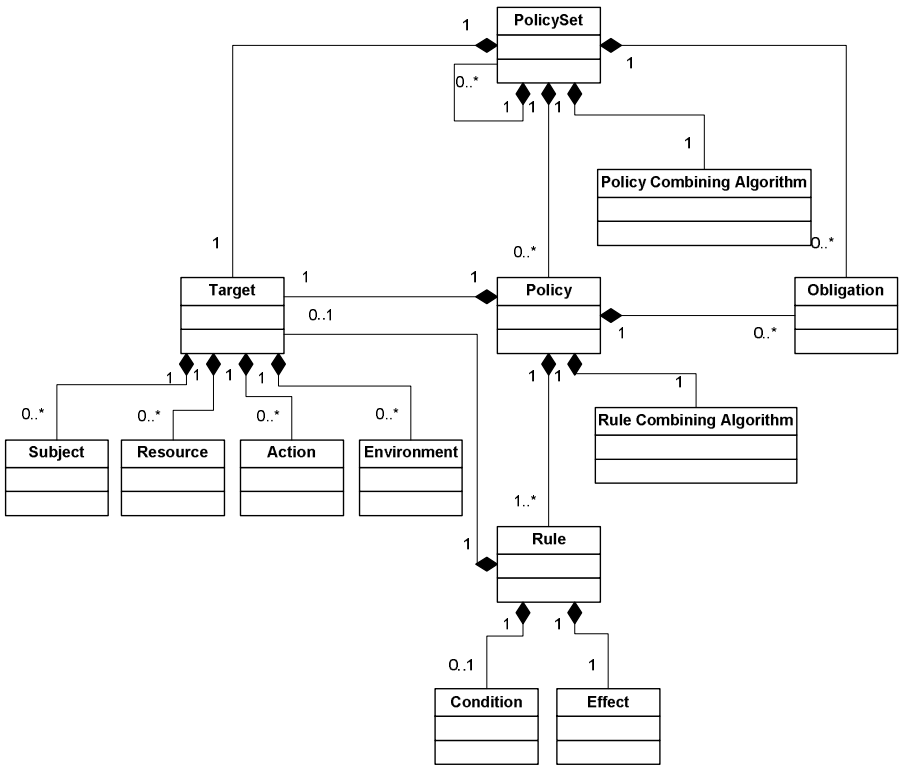


Figure 2.6: XACML Policy Language Model

"Permit" or "Deny". An XACML PDP finds a policy that applies to a given request by comparing the *Target* conditions in a *PolicySet*, *Policy* or *Rule*. A *Target* basically defines a set of conditions for the *Subject*, *Resource*, *Action* and *Environment* that must be met to a given request. Once a *Policy* or *Rule* has been found and verified to apply to a request, the *Rule*'s *Condition* is evaluated, then the *Rule*'s *Effect* is returned as the access control decision with successful evaluation of the *Rule*. Because a *Policy* or *PolicySet* may contain multiple *Policies* or *Rules*, each of which may evaluate to different authorization decisions, XACML introduces a collection of *Combining Algorithms* to combine multiple decisions into a single decision. There are *Policy Combining Algorithms* used by *PolicySet* and *Rule Combining Algorithms* used by *Policy*. An example of these is the *Permit-overrides* algorithm, which returns a "Permit" decision when any individual results of evaluation is "Permit". Figure 2.6 illustrates the overall policy language model of XACML [84].

2.2.4 Security Assertion Markup Language (SAML)

The Security Assertion Markup Language (SAML) is an OASIS standard language and protocol for exchanging authentication and authorization information between security domains [82, 78]. SAML allows business entities to make SAML assertions regarding a subject to relying parties. Each SAML assertion may encapsulate a set of statements pertaining to a particular category (*Authentication*, *Attribute* or *AuthorizationDecision*) into a standard XML structure. Each assertion holds metadata such as the issuer identity, assertion identifier, and protocol version numbers as well as conditions and advice. Assertion validity dates are a specific form of a condition. And other standard condition definitions address caching and intended audience restrictions.

Three types of statements are provided by SAML:

- Authentication statement: An authentication statement asserts that a principal did indeed authenticate with the asserting party at a particular time using a particular method of authentication.
- Attribute statement: An attribute statement asserts that a subject is associated with certain attributes. An attribute is simply a name-value pair. Relying parties use attributes to make access control decisions.
- Authorization decision statement: An authorization decision statement asserts that a subject is permitted or denied to perform certain action to specific resource.

In the SAML message exchange protocol, an entity that makes SAML assertions is called an *Identity Provider*, and the relying party that consumes SAML assertions for security management purposes is called a *Service Provider*. A SAML protocol is a simple request-response protocol, where a service provider sends a SAML *Request* element to an identity provider requesting certain SAML assertions, and the identity provider returns a SAML *Response* element to the requester with the SAML assertions being requested. SAML assertions can be bound to many different underlying communications and transport

protocols such as HTTP POST requests or XML-encoded SOAP messages. Many security frameworks, such as the Liberty Alliance [3], the Internet2 Shibboleth project [25], and OASIS Web Services Security (WS-Security) [81], have all adopted SAML as an underlying technology enabling cross-platform security management communications.

2.3 Current State of Security Systems in Collaborative Infrastructures

2.3.1 Grid Security Mechanisms

The open source Globus Toolkit [12] is the core Grid infrastructure that provides all necessary functionalities for running Grid jobs including resource monitoring, discovery, and management. Inside the current release of Globus Toolkit version 4 (GT4), the Grid Security Infrastructure (GSI) [47, 113] is implemented as the de-facto Grid security standard for authentication and message protection. GSI utilizes standard X.509 public key certificates (PKCs) [58], public key infrastructure (PKI) [6], the SSL/TLS protocol [61], and X.509 Proxy Certificates [110] as its core components. In particular, an important requirement for authentication in Grids is to support single-sign-on, which allows a user to authenticate once and gain access to the resources of multiple software systems within a Grid VO without being prompted to log in again. In addition, it is often the case that a Grid user needs to delegate a subset of his/her privileges to another entity or process for a brief amount of time. For example, a user who needs to move a large dataset for experiments may want to grant to a reliable file transfer service to access the dataset and perform a set of file transfers on the user's behalf. Proxy Certificates [110] allow an entity holding a standard X.509 PKC to delegate its identity or privileges to another entity, so that the bearer may authenticate and establish secured connections with other parties in the same manner as a normal X.509 end-entity certificate. Proxy Certificates share the same format as X.509 PKCs, binding a unique public key to a subject identity. Meanwhile, Proxy Certificates also explore the legitimate extension fields of X.509 PKCs for carrying necessary policy statements to constrain the privilege delegations.

In terms of authorization, GSI elaborates a proprietary access control list (ACL) type of policy as located in a *gridmap* file. A *gridmap* file specifies mappings of a user's global identity in his/her X.509 PKC (called distinguished name, DN) to a local account of the resource operating system. Users are authorized to use the resources when their DNs appear in such list and the privileges are determined by the associated local account. This allows the GSI to be layered securely on top of existing systems and to provide uniform credentials and certification infrastructure without undermining local security mechanisms. This method of security, however, is very coarse-grained and does not scale to ascertain the privileges for a large number of Grid users. Many alternatives have been proposed in the Grid community.

The CAS [92] framework from Globus Alliance [53] builds on the GSI infrastructure to support group authorization. It segregates the administration of resources from the administration of Grid communities. Every Grid community instantiates a CAS server controlled by a community administrator. The community administrator is trusted by all resource providers in the community to manage authorization permissions among the community users. A community user obtains an X.509 Proxy Certificate as his/her individual credential from the CAS server indicating the authorizations being assigned to him/her, and such credential is then presented to the resources in the community to gain the access. Another community-based authorization framework is realized in VOMS system [11]. VOMS and CAS are similar in which both implement a *push* architecture where the authorization servers issue policy assertions to a user, then the user presents the assertions to obtain VO issued rights. However, the VOMS-based system represents privileges in the form of X.509 ACs rather than X.509 Proxy Certificates. Another difference is that VOMS server asserts the memberships of a user, and it is the responsibility of each resource provider site to decide the rights granted to the user with those asserted memberships, while CAS grants a user's privileges centrally without any further interpretation by the resource site.

The Akenti [108] system enforces access control on resources based on policies expressed by multiple distributed stakeholders. Akenti provides a policy language to define infrastructure components for access control policies. Akenti makes an extensive use of X.509 PKCs as the authorization token for encoding both user attributes and usage conditions. Akenti allows the certificates to be stored in distributed remote repositories and provides mechanisms based on the *pull* architecture to ensure that all applicable usage conditions are retrieved and combined when making an access control decision. However, Akenti, as a stand alone authorization system has not been fully explored to use with Grid applications. PERMIS [27], on the other hand, provides a Privilege Management Infrastructure (PMI) [64] that leverages the role based access control. The RBAC policy, in a self-defined XML format, is used to control access to all resources within the policy domain and is composed of a number of sub-policies. In PERMIS, policies and user attributes are held in X.509 ACs. PERMIS has recently been integrated with GT4 infrastructure to serve as an internal or external authorization service for Grid resources.

2.3.2 P2P Security Mechanisms

P2P networking has been popularized by grassroots, mass-culture (music) file-sharing and highly parallel computing applications that may scale to hundreds of thousands of anonymous peer nodes. SciShare [18, 7] is among the few P2P-based information sharing systems that provide access control mechanisms. SciShare leverages X.509 PKCs to achieve fully distributed membership and access control at each peer agent. However, the access management in SciShare remains in a primitive group-based discretionary access control (DAC) approach that each peer agent maintains an ACL of authorized peers and resources. Waste [111] is a secure file-sharing system that provides information within a small trusted group of peers. It secures all communications at the transport-level using TLS [61] and builds a PKI web of trust [96] between peers. Waste assumes that all of the trusted peers are equal, thus any peer that is allowed into the system has full access to all information in the system. LionShare [76, 75] is an academic-oriented P2P system to share academic ma-

terials among users at many different academic institutions. LionShare utilizes PKI based authentication mechanisms and DAC based ACLs to protect resources. None of these solutions provide adequate fine-grained access control for the resources being shared in the collaboration.

Summary: In this Chapter, we reviewed and analyzed related access control models, security management mechanisms and security systems for collaborative environments, which we were motivated to propose an innovative approach to supporting secure information sharing in ad-hoc collaboration. In Chapter 3, we clearly define the ad-hoc collaboration problem domain and analyze the unique access control requirements as well as the trust management concerns associated with the environment.

CHAPTER 3: PROBLEM DOMAIN ANALYSIS

In this chapter we proceed with a concrete scenario in the context of public health surveillance where collaboration and information sharing are critical to achieve early detection of outbreaks. From the scenario, we first identify the generic access control requirements and trust management concerns associated with ad-hoc collaboration, and then analyze the patterns of information sharing and dissemination in ad-hoc collaboration.

3.1 Ad-hoc Collaboration Scenario and Access Control Requirements

Public health and surveillance data are collected by various laboratories, health care providers, pharmacies, and government agencies at local, state, and national levels. As shown in Figure 3.1, the public health surveillance requires immediate collaboration and information sharing among these sources for real-time outbreak detection and monitoring. Abnormality at any point would simultaneously trigger the establishment of collaborations.

Suppose Regional Medical Center (RMC) receives a dramatic increase in the number of patients with suspicious flu-like symptoms. This could be a sign of

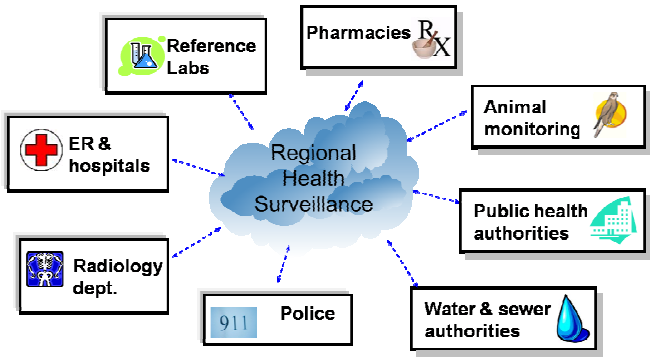


Figure 3.1: Ad-hoc Collaboration Scenario

new disease breakout or an incident of bioterrorism attack. The medical data need to be shared immediately with collaborating laboratories and specialities to identify the causing virus or bacteria. Professionals in other hospitals and clinics may also need to share the information for more effective diagnosis and treatments. In addition, regional public health authorities have to be notified to investigate and track the spread of the epidemics, where the surveillance information may further be disseminated to state and federal level agencies.

However, sharing of sensitive medical data is restricted by privacy protection regulations and federal laws. The owner of the data records (RMC) is responsible to apply appropriate conditions restricting the access and dissemination of the data. As multiple agencies and institutes may involve in the collaboration with different duties, RMC has no priori knowledge on who exactly and in which agencies the users are going to access the data. It has posed great challenges for RMC to apply effective control on the data access and dissemination among all these remote users.

From the scenario above, we first differentiate the concepts of *collaborating organizations* and *collaborating users*. In our research, we assume that the resource owner (also called *originator*) has limited trust on some collaborating organizations based on pre-established relationships. The collaborating organizations do not directly share the data. Instead, they carry out certain responsibilities to identify and nominate users from their perspective domain. It is the individual users nominated by these organizations who actually access to the data, and we call these users as *collaborating users*. In our scenario, *RMC* is the originator who is responsible to disseminate the surveillance data to other parties in the collaboration. The laboratories, government bureaus, relevant clinics and pharmacies are collaborating organizations, while the users within these organizations (i.e., lab investigators, government staffs, doctors, assistants and pharmacists) are collaborating users who need to access and/or disseminate the data to conduct their duties. As shown in the sce-

nario, the collaboration is always triggered "on the fly" by sudden events. Ensuring secure and authorized information sharing in such dynamic and spontaneous environment is a challenging task.

- **Originator control:** The "on the fly" nature contributes to frequent changes in the scope and structure of the 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. Therefore, entities towards the data resource can then 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. An originator should have the ultimate authority over its data resource to clearly define who is authorized to access the data and to what extent the data information can be distributed. We refer to the scope of information dissemination as the originator's *collaborative sharing control domain*.

- **Flexible and manageable access control:** Since ad-hoc collaboration may involve a large number of distributed collaborating users, an originator lacks the knowledge on who exactly needs access to the data. It is impossible for an originator to explicitly specify authorizations based on users' identities. Instead, abstractions are required to manage unknown collaborating users and authorize their privileges in a flexible and manageable way. More likely, collaborating 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 ad-hoc collaboration, an originator is responsible to define its own trust relationships and delegate authorities among distributed collaborating parties. In the paradigm of attribute-based access control, user attributes may

be asserted by various attribute authorities, and these attribute authorities may not be trusted by an originator to the same extent. For instance, a user may claim his/her residence by presenting either a passport issued by US government or a driver's license issued by a local DMV office. The originator may only trust the passport in some cases. Therefore, the criteria to determine the trustworthiness of user attributes should also be specified as part of the access control requirements [68].

- **Effective dissemination control:** Digital information sharing involves data transmission among participating parties. In order for an originator to maintain the control power when information leaves the originator's domain, a policy-driven approach with distributed policy propagation and enforcement scheme is required in the sharing infrastructure. Ideally, an originator should be able to specify access management policies, and it is the responsibility for the authorization infrastructure to enforce these policies on the originator's behalf along with the information dissemination.
- **Fine-grained selective sharing:** The resource being shared in ad-hoc collaborations may be a composition of different types of sub-objects. Such resources need to be shared fully or partially depending on different collaboration purposes and degrees of sensitivity. As a special requirement in our motivation scenario, the medical data collected by the originator *RMC* involve a complex composition of sensitive information, including patient demographic details, medical history, examination reports, laboratory test results, radiology images (X-rays, CTs), and so on. There is a strong need for protection models to comply with legal and regulatory policies, while simultaneously ensuring that access to sensitive information is limited only to those entities who have a legitimate need-to-know requirement.

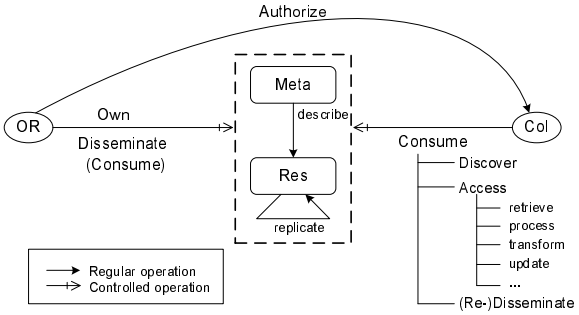
3.2 Collaborative Sharing Patterns and Dissemination Requirements

The originator and the collaborating user are two essential actors involved in the information sharing process with different responsibilities and privileges. As shown in Fig-

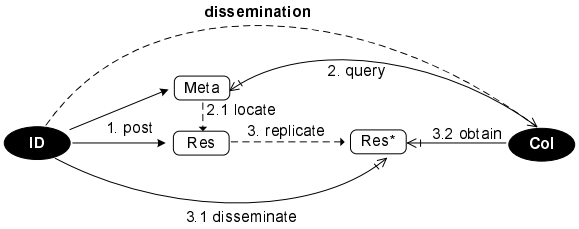
ure 3.2(a), an originator (*OR*) has the ownership privilege to the data resource and is responsible to make the data available in the collaboration. The collaborating users (*Col*), on the other hand, are the consumers of the resource, whose sharing activities should be authorized by the originator.

Despite the variety of supporting infrastructures (i.e., P2P and Grid computing), collaborative information sharing always involves a sequence of generic activities between these two actors including *resource publication*, *resource discovery*, *resource access* and *resource dissemination/redissemination*. The *resource publication* starts with an originator making the data resource available in the collaborative community along with certain metadata information that describes the resource to facilitate the resource discovery. A remote collaborating user locates the resource in the process of *resource discovery* by querying certain desired properties of the resource. The user can further request to share the resource by either directly accessing (*resource access*) or indirectly obtaining a copy of the data resource (*resource dissemination*). With an originator's authorization, a collaborating user may further re-disseminate the data resource to other legitimate users. In the perspective of resource dissemination, we consider the originator as an *initial disseminator* (*ID*) as it 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* (*DD*). Figures 3.2(b) and 3.2(c) illustrate the use cases as stepwise procedures for resource dissemination and re-dissemination. With the requirement of originator control, an originator is responsible to authorize collaborating users based on each identified step and the access control system should effectively enforce the originator's policies throughout the whole information sharing process.

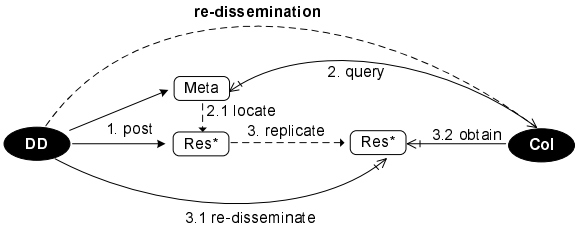
In order to control the scope and direction of information dissemination, a virtual *collaborative sharing domain* towards a specific data resource should be clearly defined by an originator to include the originator itself as the initial disseminator (*ID*) and a set of collaborating users (*Col*) as intended recipients, among which some are promoted as des-



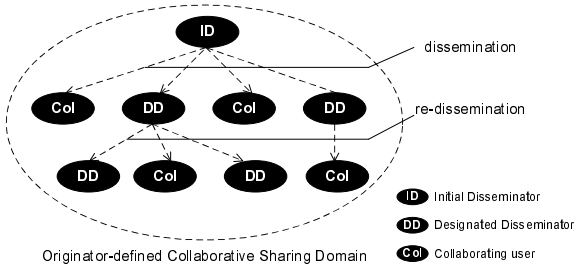
(a) Actors and General Activities



(b) Resource Dissemination



(c) Resource Re-dissemination



(d) Overall Collaborative Sharing Pattern

Figure 3.2: Use Patterns of Resource Dissemination and Re-dissemination

ignated disseminators (*DD's*). Figure 3.2(d) shows a well-defined and highly regulated information dissemination pattern within an originator's collaborative sharing domain. In particular, only *ID* and *DD's* could distribute the information to other legitimate *Col's*. 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's* and the information flows coming from *DD's* belong to the type of re-dissemination. All other *Col's* inside the domain form the leaf nodes of the sharing tree. The originator's collaborative sharing domain is a virtual control domain to restrict the scope of information dissemination, and it should be independent of any physical organizational settings. Therefore, it is essential to have a flexible yet effective way for an originator to define such virtual domain and regulate the information dissemination flows among collaborating users.

Summary: In this Chapter, we started with a concrete collaboration scenario in public health surveillance to identify the generic access control and trust management requirements associated with ad-hoc collaboration. We further derived the unique dissemination control requirements for collaborative information sharing by analyzing information sharing patterns within the environment. In Chapter 4, we formally articulate the identified access control requirements and elaborate the design of RAMARS framework to meet these requirements.

CHAPTER 4: RAMARS ACCESS MANAGEMENT FRAMEWORK

4.1 General Principles

As reviewed in Chapter 2, Role-based access control (RBAC) provides great advantages in reducing the complexities of security management in large-scale and enterprise-wide systems by using the notion of role as the central construct to abstract users and privileges. In an ad-hoc collaboration environment, an originator could indirectly define its collaborative sharing domain and delegate information sharing capabilities through 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. Accordingly, we define the following two types of roles and introduce a special reference relationship between the roles to address the permission-role assignment.

- **Normative sharing roles:** These roles need to be accomplished by all collaborative sharing services 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.

- **Originator roles and collaborator roles**: These roles are introduced to reflect the special characteristics of the two major actors: the originator and collaborating users. Especially, an originator could abstract its collaborative sharing control domain through a set of collaborator roles. As the information sharing relationship is based on specific data resource, the set of collaborator roles should be specified on per data resource basis. For different data resources, an originator could define different collaborative sharing domains including different sets of collaborator roles. Through this approach, an originator could effectively tune the intended scope of data dissemination to accommodate different collaboration and information sharing purposes.
- **Role reference:** As another extension to the traditional RBAC permission-role assignment, our framework allows an indirect permission assignment as *role reference*, where a collaborator role is mapped to a certain normative sharing role through which to exercise information sharing capabilities.

Introducing roles effectively reduces the management complexity for an originator to maintain its collaborative sharing domain, yet the user-role assignment remains as an issue. 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 assigned to collaborator roles based on their attributes instead of identities. These attributes could be general information about a user's name, address and date of birth, or more relevant information such as the user's security clearance, affiliation, and qualification. Different from most trust management approaches [99, 22, 73], user attributes do not directly associate with authorizations, but serve as a profile of the user to gain the collaborator role membership within the originator's collaborative sharing domain.

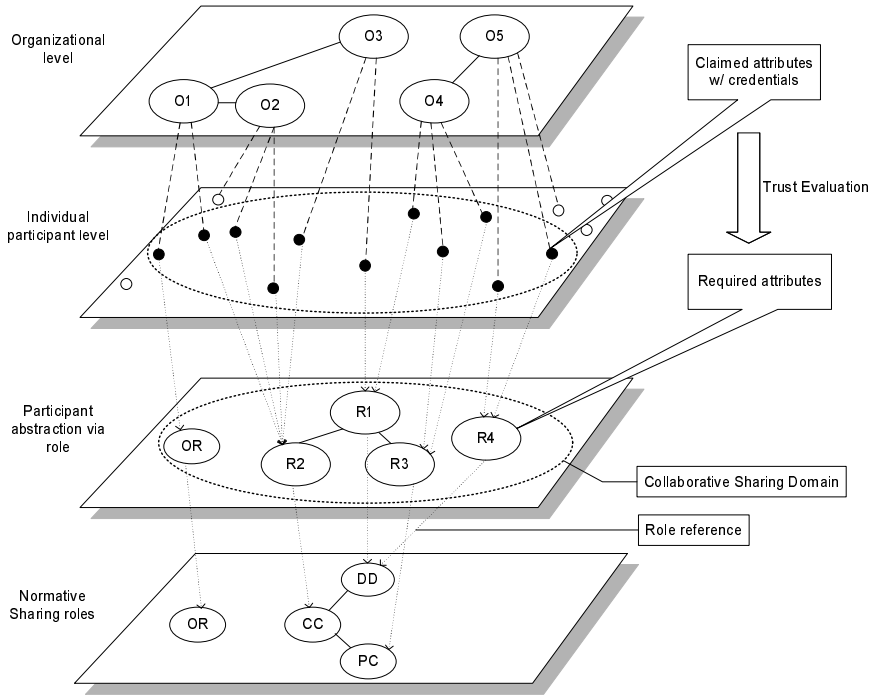


Figure 4.1: RAMARS Approach Illustration

In addition, as the set of collaborator roles conveys an originator's collaborative sharing domain, the originator has the ultimate localized authority over these roles, and such authority is never delegated. Instead, without assuming any centralized trust base, a special type of *delegation of authority* is introduced for an originator to delegate attribute authorities with different degrees of trust. An originator may define different sets of required attributes for remote users to be assigned to certain collaborator role(s), and collaborating organizations are delegated to nominate users with such attributes from their perspective domains. By changing the sets of required attributes and delegation relationships, the unknown collaborating users could be dynamically assigned to different collaborator roles to exercise various information sharing capabilities. A collaborating user should claim the possession of required attributes by presenting credentials issued by certain attribute authorities. Yet it is up to the originator's discretion to determine the trustworthiness of these credentials, thereafter to decide whether the claimant of such attributes can be accepted for the role assignment. We therefore introduce a special type of *trust management constraint*, associ-

ated with a set of evaluation policies and procedures, to determine the trustworthiness of user attributes. In doing so, we embed the trust-aware feature to the dynamic role assignment based on user attributes. Figure 4.1 summarizes the overall role-based approach with trust management feature conveyed in our RAMARS framework to achieve the effective originator control.

4.2 Formal RAMARS Model

We propose a framework called Role-based Access Management for Ad-hoc Resource Sharing (RAMARS) that contains substantial extensions to traditional RBAC models. In this Section, we define a collection of basic elements and relations that covers the core set of features in RAMARS to achieve effective access management in collaborative resource sharing systems.

All entities involved in the collaboration (i.e., organizations, institutes, individual users, etc.) are generalized as E, and the collaborating users (U) are the individual users who need to be authorized to gain access to the data. Each collaborating user is associated with collections of attributes (AS), and we call such association relationship as a *user-attributes entitlement* denoted as ETL. Different user-attributes entitlements serve as different profiles of a user for authorizations. As we do not assume that an originator accepts all attributes claimed by a collaborating user, a special trust management (TM) constraint is in place to determine the trustworthiness of user attributes. The details of TM constraint is discussed in Section 4.4. In terms of roles, RAMARS contains the aforementioned normative sharing roles ($NSHR$), originator roles (ORR) and collaborator roles (COL). The partition of originator and collaborator roles induces a parallel partition of role assignment. The entity who owns the data resource, organizationally or individually, is assigned to the originator role through a relation of UAO. The collaborating users, through their trusted user-attributes entitlements, are assigned to collaborator roles in a relation of UAC. The identified sharing activities/operations (OPN) are assigned to normative sharing roles through a relation of $CAPN$. And role hierarchy (RHN) is also introduced to realize the capability dependen-

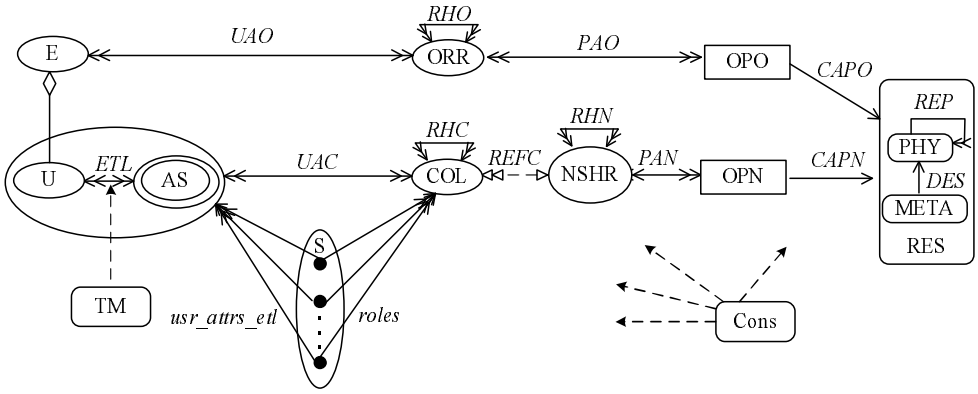


Figure 4.2: RAMARS Model

cies among the normative sharing roles. Collaborator roles achieve information sharing privileges through an indirect role-role reference relation as $REFC$. Finally in RAMARS, the data resource is further separated into metadata ($META$) and physical resource (PHY). Even though we deal with digital information, we use the term "physical" to distinguish the actual data resource from its metadata. The physical resource can be replicated during information dissemination (REP). The structure of RAMARS model is summarized in Figure 4.2. We organize the definitions of our model in a number of logical parts, similar to the ones articulated in RBAC models, such as roles, users, resources (objects), capabilities (permissions), and sessions.

Roles, Role Hierarchies and Role References.

Let ORR, COL and $NSHR$ be the sets of originator roles, collaborator roles, and normative sharing roles, respectively. We define:

- $RHO \subseteq ORR \times ORR$, partially ordered role hierarchy for originator roles.
  $RHC \subseteq COL \times COL$, partially ordered role hierarchy for collaborator roles.
  $RHN \subseteq NSHR \times NSHR$, partially ordered role hierarchy for normative sharing roles.
- $REFC \subseteq COL \times NSHR$, a many-to-one collaborator-to-normative role reference relation.
- $refers\_to(col : COL) \rightarrow NSHR$, a function mapping a collaborator role col in

COL to a normative sharing role. Formally,

$refers\_to(col) = \{nr \in NSHR | (col, nr) \in REFC\}$.

Entities and Users.

- $E = \{e_1, \ldots, e_n\}$, a set of entities to generalize all related parties (individuals and/or organizations) in the collaborative environment.

  $U \subseteq E$, a subset of individual users who need to be authorized to access to the resource.

  $ATTR = \{attr_1, \ldots, attr_n\}$, a set of attributes, and each $attr_i$ ($i \in [1, n]$) is represented as an $(Aname, Value)$ pair.

- $AS = \{as_1, \ldots, as_s\}$, a collection of attribute sets associated with individual users U, where $as_i = \{attr_i, \ldots, attr_t\} \subseteq ATTR$, $i \in [1, s]$, and s denotes the number of user profiles used for authorization.

  $ETL \subseteq U \times AS$, a many-to-many user-attributes entitlement relation, indicating the possession of user attributes.

  $etls(u : U) \rightarrow 2^{AS}$, a function mapping a user u in U to all attribute sets he/she is entitled to. Formally, $etls(u) = \{as \in AS | (u, as) \in ETL\}$.

- $UAO \subseteq E \times ORR$, a many-to-many entity-to-originator role assignment relation.

  $UAC \subseteq ETL \times COL$, a many-to-many user-attributes entitlement to collaborator role assignment relation.

- $role\_etls(col : COL) \rightarrow 2^{ETL}$, a function mapping each collaborator role col in COL to a set of user-attributes entitlements in the presence of a role hierarchy. Formally,

  $role\_etls(col) = \{etl \in ETL | (\exists col' \succeq col)[(etl, col') \in UAC]\}$.

Resources.

Let RES (PHY and $META$) be a set of resources including physical resource set and metadata set. We define:

- $RES = PHY \cup META$.
- $DES \subseteq META \times PHY$, a one-to-one resource description relation.
- $REP \subseteq PHY \times PHY$, a one-to-many resource replication relation.
- $replicate(phy : PHY) \rightarrow 2^{PHY}$, a function mapping an original resource phy in PHY to a set of replicated resource copies. Formally,
  $replicate(phy) = \{phy' \in PHY | (phy, phy') \in REP\}$.

Capabilities.

Let OPO and OPN be sets of operations for originator roles and normative sharing operations, respectively. We define:

- $CAPO = 2^{(OPO \times RES)}$, a set of capabilities for originator roles.
  $PAO \subseteq CAPO \times ORR$, a many-to-many capability-to-originator role assignment relation.
- $CAPN = 2^{(OPN \times RES)}$, a set of normative capabilities.
  $PAN \subseteq CAPN \times ORR$, a many-to-many capability-to-normative sharing role assignment relation.
- $cap\_n(nr : NSHR) \rightarrow 2^{CAPN}$, a function mapping a normative sharing role nr in $NSHR$ to a set of normative sharing capabilities in the presence of a role hierarchy. Formally,
  $cap\_n(nr) = \{cap \in CAPN | (\exists nr' \preceq nr)[(cap, nr') \in PAN]\}$.
- $cap\_o(or : ORR) \rightarrow 2^{CAPO}$, a function mapping an originator role or in ORR to a set of originator specific capabilities in the presence of a role hierarchy. Formally,
  $cap\_o(or) = \{cap \in CAPO | (\exists or' \preceq or)[(cap, or') \in PAO]\}$.
- $capabilities\_c(col : COL) = cap\_n(refers\_to(col))$, a function indirectly mapping a collaborator role col in COL to a set of normative sharing capabilities through reference.

Sessions.

Let S be a set of sessions. We define:

- $usr\_attrs\_etl(s_i : S) \rightarrow ETL$, a function mapping each session s_i in S to a single user-attributes entitlement.
- $roles(s_i : S) \rightarrow 2^{COL}$, a function mapping each session s_i in S to a set of collaborator roles in the presence of a role hierarchy. Formally, $roles(s_i) = \{col \in COL | (\exists col' \succeq col)[(usr\_attrs\_etl(s_i), col') \in UAC]\}$.

4.3 An Extended Example

Before proceeding to introduce the trust-aware feature conveyed in TM 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 ABC Community Hospital (ABC) is a registered institute in the public health surveillance scenario. The Emergency Care Center (ECC) in ABC hospital needs to share RMC's medical information for patient treatment. RMC requires that an individual user must be a US citizen according to the security clearance guidelines, and he must be a member of ECC in ABC hospital. In order to expedite the data dissemination in emergency situations, RMC allows the chair of ECC to re-disseminate the data to relevant on-duty employees. Suppose Dr. John Doe is in charge of ECC 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 then responsible to assert Dave's affiliation

and department membership on behalf of ABC. And finally, Dr. John Doe as the chair of ECC could confirm that Dave is an on-duty physician assistant to share the information with RMC.

In this example, *RMC* is the *originator* of the medical data, and each individual member in *ECC* of *ABC* hospital (i.e., *John* and *Dave*) is considered as a *collaborating user* who needs to be authorized to access and/or disseminate the data. In terms of information sharing operations, we consider very simple ones as discussed in Section 3.2: $CAPN = \{query, obtain, post, (re)disseminate\}$. Accordingly, there are three normative sharing roles associated with these sharing operations: *Designated Disseminator* role (*DD*), *Common Collaborator* role (*CC*) and *Potential Collaborator* role (*PC*). In particular, *query* is the most fundamental resource discovery capability associated with *PC* role, *obtain* is an advanced resource access capability of *CC* role with *query* as a prerequisite, *post* and $(re)disseminate$ are the most advanced dissemination capabilities associated with *DD* role. Therefore 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 its collaborative sharing domain, 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 the medical information. The collaborating user who is a member of *HCP* role can *obtain* the data resource, but cannot further distribute the data resource. Yet a user with *Coordinator* role could redistribute the data to other legitimate users within *RMC*'s collaborative sharing domain. In our example, all users within *ABC* must be US citizens and work in *ECC*. In addition, *John* as the *chair* of *ECC* should be assigned to the *Coordinator* role, so that he is able to re-disseminate the data. *Dave*, on the other hand, as an on-duty physician assistant (*PA*), is assigned to the *HCP* role for data access. Figure 4.3 summarizes the example using concepts introduced in RAMARS framework and lists several sample credentials presented by *John* and *Dave* to claim their attributes. When *Dave* tries to access

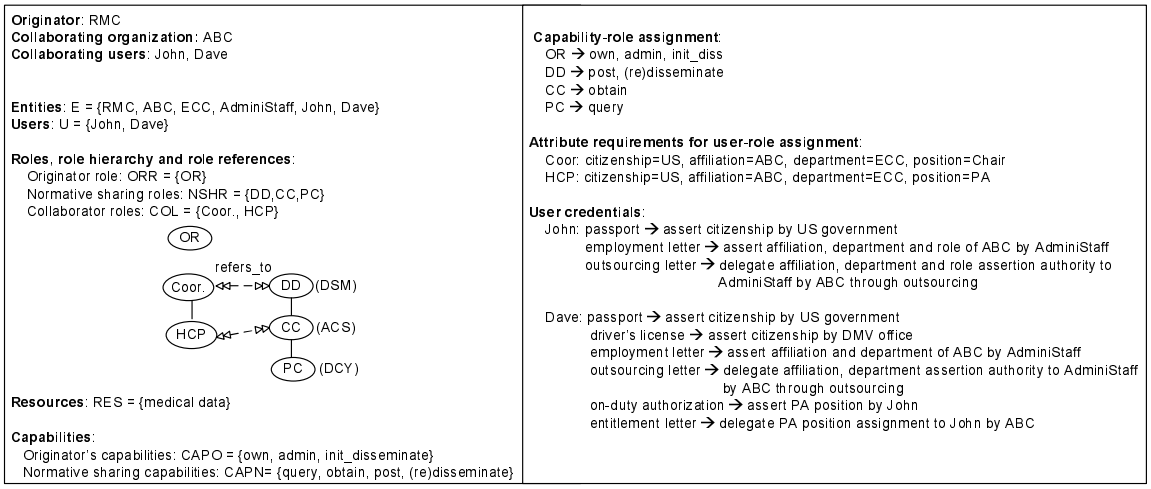


Figure 4.3: Collaborative Sharing Example

the data resource, how could the originator *RMC* evaluate the trustworthiness of *Dave*'s attributes based on his credentials? The issue is handled by the *TM* constraint.

4.4 RAMARS in Trust Management Layer – *TM* Constraint

TM constraint is an important component in RAMARS for an originator to determine the trustworthiness of a user's attributes based on his/her credentials. An attribute (*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* of a user's attribute is a specific value within the attribute value domain or null, formally, $value \in domain(AName) \cup \{null\}$. In simplicity, we denote an attribute as an *(AName, value)* pair. In a collaborative community, we assume that there is a fixed set of attribute names agreed by all the participating parties.

A collaborating user claims an attribute by presenting supportive **credential(s)**. 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 (certifier). 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. In our example, *ABC* delegates two attributes of (affiliation,ABC) and (department,ECC) to *AdminiStaff* through outsourcing. Many often, a credential is associated with certain validation constraints, such as the validity period and maximum delegation depth [9]. We use the notion of *context* to abstract these constraints that may be used to determine the validity of a credential. And each validation constraint in *context* can be identified through a *type* indicator. For instance, given context constraints Ctx of a credential, Ctx_T indicates the constraint of the validity time period, and Ctx_D indicates the constraint of the maximum delegation depth. To summarize, we define a credential as follows:

Definition 1 *(***Credential***).* A credential is defined as a tuple of $cred = (holder, attrs, certifier, Ctx)$, where $holder, certifier \in E$, $attrs \subseteq ATTR$ is the asserted user attributes, and Ctx is the validation constraints. We use the dot notation to refer to the elements in a credential, such as $cred.holder$.

When the delegation of attribute authority is considered, a chain of attribute assertion can be constructed to realize the trust propagation extending from the initial attribute authority to the end user via several intermediate delegated attribute authorities. We refer this chain as an **assertion path**. User attributes can be asserted through multiple assertion paths, for instance, directly asserted by an attribute credential, and/or indirectly asserted by chains of delegations. In our example, the passport realizes a direct assertion path with one attribute credential to assert the attribute of (citizenship,US) for *Dave*, while the delegation credential issued by *ABC* and the attribute credential issued by *AdminiStaff* realize an indirect path of *ABC*→*AdminiStaff*→*Dave* to assert the attributes of (affiliation,ABC) and (department,ECC).

Definition 2 *(***Assertion path***).* An assertion path $ap[attrs]$ is defined as a sequence of credentials $ap[attrs] = cred_1 cred_2 \ldots cred_n$, where $attrs \subseteq ATTR$, $cred_i.attrs = cred_{i+1}.attrs$, and $cred_i.holder = cred_{i+1}.certifier$ for all $i \in [1,(n-1)]$ and

$n \geq 1$. We also define a function to retrieve the number of credentials in an assertion path: $depth(ap[attrs]) = |ap[attrs]|$. Especially,

- if $depth(ap[attrs]) = 1$, $attrs$ is directly asserted by an attribute certificate;
- otherwise, $attrs$ is indirectly asserted through a delegation chain.

As the same attribute can be possibly asserted by multiple assertion paths, we further define a function of $num\_aps(attrs)$ to capture the total number of assertion paths asserting same $attrs$. For example, Dave's passport and driver's license establish two direct assertion paths for his citizenship attribute. It can be denoted as: $num\_aps(\{(citizenship, US)\}) = 2$.

Utilizing assertion paths establishes a means for an originator to build trust relationships with remote collaborating users through several delegated mediators. The first step towards the trust evaluation then is to check the validity of an assertion path. We define the following validation function for the evaluation. Any invalid assertion path with a *false* value being returned by the evaluation function should be discarded before any further trust evaluations.

Definition 3 *(***Assertion path validation function***).* Let $ap[attrs] := cred_1 cred_2 \ldots cred_n$ be an assertion path. The validation function is defined by a boolean function:

$validate\_ap(ap[attrs], EN) \overset{cred.Ctx^*}{\rightarrow} \{true, false\}$, where $ap[attrs]$ is the assertion path to be validated, $cred.Ctx^* \sqsubseteq cred_1.Ctx \otimes cred_2.Ctx \otimes \cdots \otimes cred_n.Ctx$ denotes the validation constraints for all credentials in the assertion path, and EN captures the environmental parameters required for the evaluation. For instance, *CurrentDate* is an environmental parameter captured for validity period evaluation.

The function evaluates the validity using the algorithm shown in Figure 4.4. In particular, each credential element in the assertion path $cred_i$ ($i \in [1, n]$) is validated against its context constraints Ctx. Any invalid credential would result in a *false* value being returned by the validation function. Otherwise, the credential is further evaluated against the delegation constraint when the position of the credential in an assertion path is considered.

```
Algorithm validate_ap
  Input: ap[attrs], EN        /* ap[attrs] is a particular assertion path to be validated, EN is the
                                 environmental parameters */
  Output: true if valid, otherwise false

/* check validity for direct assertion */
IF depth(ap[attrs]) = 1 THEN     /* if there's only one attribute credential in the assertion path */
    result = validate_cred(cred_i, EN);
    return result;                    /* validate the credential and return the result */

/ * check validity for indirect assertion */
ELSE
    FOR each (credi∈ap[attrs]) DO
        result = validate_cred(cred_i, EN);
        IF result = false THEN
            return false;             /* validate each individual credential */
        ELSE IF credi.Ctx.D< (depth(ap[attrs])-i) THEN
            return false;             /* validate the depth against delegation constraints */
return true;
```

Figure 4.4: Algorithm for Assertion Path Validation Function

Neither the position of a credential in an assertion path, nor the total depth of an assertion path can exceed the maximum delegation depth defined in its delegation constraint.

Given multiple assertion paths for certain user attributes, the attributes may not be trusted with the same degree by an originator. Therefore, the originator should have its own interpretation on the degree of trustworthiness for the user attributes. We thus introduce a notion of trust level, defined as a partial order $(TL, \preceq)$, for an originator to subjectively rank and compare different assertion paths for the user attributes. We suggest the following three major factors for an originator to determine the trust level: (1) the certifier of the credential; (2) the depth of an assertion path extended to an end user; (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. The attributes asserted by different collaborating organizations result in different trust levels. In addition, an assertion path is considered as a way for the originator to propagate trust to a remote user. Both the delegated collaborating organizations and the steps of delegation may vary the degree of trust. 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 [96]. These factors, individually or combined together, may contribute to the trust level assessment. A trust assessment policy then should be in

Table 4.1: Function Specifications for TM Evaluation

Functions	Semantics
$validate\_ap(ap[attrs], EN) \overset{cred.Ctx^*}{\rightarrow} \{true, false\}$	Return *true* if the assertion path for *attrs* is valid.
$trustAssessment(attrs, aps) \overset{TAP.TL}{\rightarrow} TL$	Return a trust level *tl* of an assertion path *ap[attrs]* given a trust assessment policy *TAP.TL*.
$trustDecision(attrs, tl) \overset{TAP.TD}{\rightarrow} \{true, false\}$	Return *true* for the asserted *attrs* if the trust level *tl* achieved by a supportive assertion path *ap[attrs]* meets the trust assessment policy *TAP.TD*.
$depth(ap[attrs]) \rightarrow N$	A utility function that returns the depth of an assertion path *ap[attrs]*.
$num\_aps(attrs) \rightarrow N$	A utility function that returns the total number of assertion paths for the same set of attributes.

place accordingly to derive a specific trust level for each user attribute under evaluation considering these trust evaluation factors. Given different trust levels achieved by a user's claimed attributes, the ultimate question for *TM* constraint is whether these attributes can be trusted by the originator to determine the user's role membership. Again, this should be expressed in the originator's trust assessment policy as well.

Definition 4 *(***Trust level assignment function***).* We define the function mapping a trust level to the asserted user attributes given a set of assertion paths: $trustAssessment(attrs, aps) \overset{TAP.TL}{\rightarrow} TL$, where $attrs \in ATTR$ is a set of claimed attributes under evaluation, aps is a set of assertion paths of the claimed $attrs$, and $TAP.TL$ is a policy used by the assessment function to derive a single trust level.

Definition 5 *(***Trustworthiness assessment function***).* We define the function of mapping the trust level of the claimed attributes to a boolean trustworthiness evaluation decision: $trustDecision(attrs, tl) \overset{TAP.TD}{\rightarrow} \{true, false\}$, where $attrs \in ATTR$ is the claimed attributes under evaluation, $tl \in TL$ is the achieved trust level, and $TAP.TD$ is a policy used by the assessment function to derive the trust decision as a boolean value.

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

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

/* Step 2: Trust level assessment */
tl = trustAssessment(attrs, APaths, TAP.TL);     /* assign trust level tl to attrs given APaths according to TAP.TL policy */

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

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

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

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

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

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

return APs;
```

Figure 4.5: Trustworthiness Evaluation Algorithm

Given a set of supportive credentials, we design an algorithm, called *evalTrust*, to evaluate the trustworthiness of the user attributes (Figure 4.5). The algorithm works as follows. In Step 1, the supportive credentials are categorized into a set of assertion paths (*APaths*) based on the asserted attributes (*attrs*). This is realized in another algorithm *findAssertionPaths*. Each path *ap* is validated using the function $validate\_ap(ap, EN)$ as defined in Definition 3. Invalid assertion paths are discarded. In Step 2, the user attributes with all valid assertion paths are evaluated against *TAP.TL*, resulting in a trust level (*tl*) being assigned to the user attributes. This procedure utilizes the trust level assessment function defined in Definition 4. Finally in Step 3, the trust level achieved by the user attributes is evaluated against the trust decision policy (*TAP.TD*) to make the final decision on whether

the user's claimed attributes are trusted for further role assignment. A value of *true* is returned if the trust level achieved by the user attributes satisfies the threshold defined in *TAP.TD*. This step utilizes the trustworthiness assessment function defined in Definition 5. And trusted user attributes are promoted to determine the user's role membership.

As a summary, Table 4.1 lists all trust evaluation and relevant utility functions in *TM* constraint, participating in a sequence of evaluation procedures as the assertion path validation, the trust level assignment, and the trustworthiness assessment. Each evaluation procedure requires a specific policy component to be available. The details of these policy components are discussed in Chapter 5.

Summary: In this Chapter, we elaborated the design and formalization of RAMARS model. RAMARS model incorporates the role-based approach and trusted attribute-based role assignment to accommodate the originator control, delegation and dissemination control requirements as we identified in Chapter 3. A special trust management constraint is introduced in RAMARS that includes a series of evaluation procedures to determine the trustworthiness of a user's attributes for the role assignment. In Chapter 5, we formally specify the policies to realize all elements in RAMARS model, and we demonstrate an implementation of the policies using standard XACML policy language.

CHAPTER 5: POLICY SPECIFICATION

5.1 Policy Components

As a policy-driven approach, we now present the policy specification to realize the elements conveyed in RAMARS model. From the requirement of originator control, an originator defines collaborator roles as its collaborative sharing domain and delegates information sharing capabilities to the normative sharing roles. An 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 to determine the trustworthiness of user attributes. In our policy framework, all these policies are conveyed inside the Role-based Originator Authorization policy set (**ROA**). As information may be disseminated among collaborating users, the system should guarantee the originator's ROA policies be enforced correspondingly along with the information dissemination. In order to facilitate such distributed enforcement of the originator's policies, we design a lightweight Root Meta Policy Set (**RMPS**) to declare the ownership of a particular resource and associate the originator's ROA policies with the data resource. By doing this, an originator can modify its ROA policies locally, and the RMPS policy is supposed to be propagated among collaborating users to guarantee the originator's ROA policies are retrieved and enforced at run time. In addition, the separation of RMPS from ROA policies also promotes the policy reusability and portability as an originator could easily associate the same set of ROA policies to different data resources for authorizations. Besides the policy sets of ROA and RMPS, the validation constraints for each credential (as represented in *cred.Ctx*) specifies the rules to restrict the validity of a credential, which also requires clear definition of their syntax and semantics. Such definition is specified in **CREDPolicy**. Our policy specification follows

Table 5.1: Policy Specification Notions and Usage

Notation	Usage
=	definition
,	concatenation
;	termination
|	separation
[...]	option
{ ... }	repetition
(...)	grouping
" ... "	double quotation marks
? ... ?	special sequence

the same notion of terminals and nonterminals as defined in the ISO standard for extended Backus-Naur form (EBNF) [4]. Table 5.1 lists the notations that we use for our policy specification based on the standard.

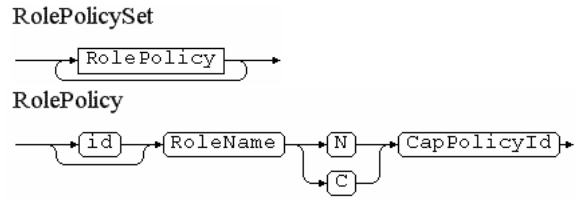
5.1.1 ROA Policy Set

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*). 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 from a *RP* to the unique identifier of the associated *CapPolicy* denoted as *CapPolicyId*. In our specification, elements such as policy *id* and *RoleName* are specified as "*?arbitrary string?*", which means an originator has the freedom to name these elements. Figure 5.1(a) shows the specification of *RPS*.

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 such capabilities by 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 via the element of *CapPolicyId_MappedRole*. With this reference, all collaborator roles can eventually be 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 generic sharing operations should be supported by all sharing infrastructures 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 authorization policies. Figure 5.1(b) shows the specification of *CPS*.

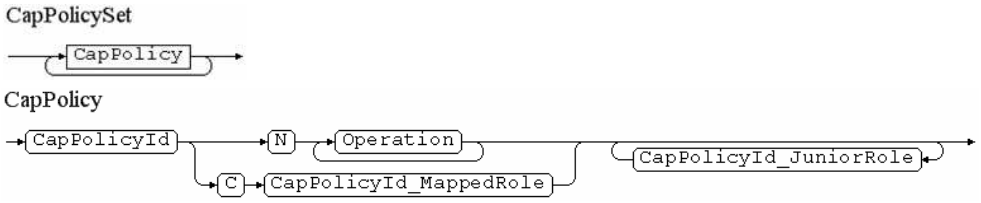
Role Assignment Policy Set (RAPS) contains one or more sub-policies (*RAPolicy*) to define the required attributes for each collaborator role. A collaborator role can be assigned according to different sets of required attributes, each denoted as a *ReqAttrGroup*. A *ReqAttrGroup* defines a set of attribute predicates mapping the values of a user's attributes




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

id            = ? arbitrary string ? ;
RoleName      = ? arbitrary string ? ;
CapPolicyId   = ? arbitrary string ? ;
```

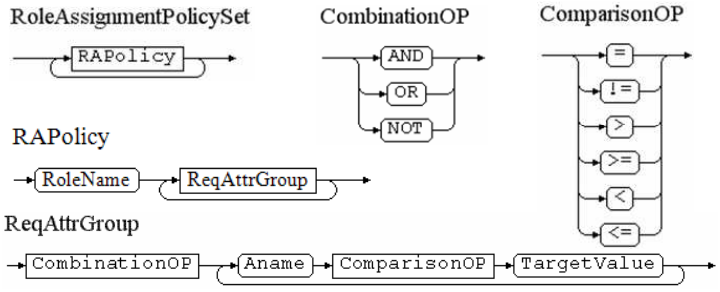
(a) ROA-RPS Specification




```
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 ;
```

(b) ROA-CPS Specification




```
RoleAssignmentPolicySet = RAPolicy, {RAPolicy} ;
RAPolicy                = RoleName, {ReqAttrGroup, {ReqAttrGroup}} ;
ReqAttrGroup            = CombinationOP, {Aname, ComparisonOP,
                          TargetValue},
                           {Aname, ComparisonOP,  TargetValue} ;
CombinationOP           = "AND" | "OR" | "NOT" ;
ComparisonOP            = "="|"!="|">"|">="|"<"|"<=" ;

RoleName                = ? arbitrary string ? ;
Aname                   = ? arbitrary string ? ;
TargetValue             = ? arbitrary string ? ;
```

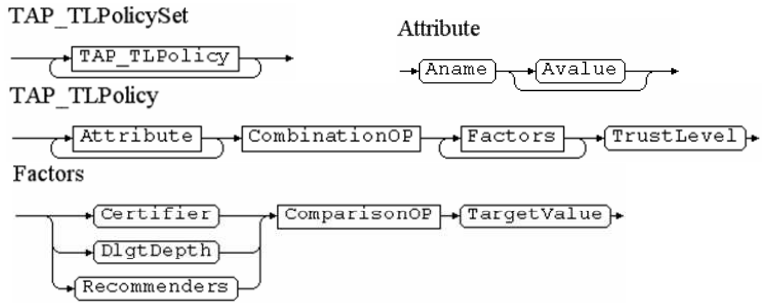
(c) ROA-RAPS Specification

Figure 5.1: ROA Policy Set Specification - Authorization

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 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 predicates must be *true*; (ii) "OR" implies that at least one attribute predicate must be *true*; and (iii) "NOT" implies that none of the attribute predicates must be *true*. The collaborator role is assigned after a *true* value is eventually derived from the *ReqAttrGroup* evaluation. Figure 5.1(c) shows the specification of *RAPS*.

Trust Assessment Policy (*TAP*) is used in two types of functionalities achieved by *TM* constraint: (i) to determine the trust level of a user's claimed attributes; and (ii) to make the trust decision on the claimed attributes. We realize these functionalities using two policy elements referred to as *TAP.TL* and *TAP.TD*, respectively.

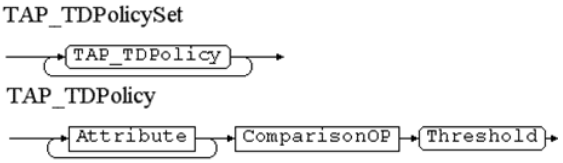
As mentioned in previous Chapter, *TAP.TL* is used for an originator to consider the affecting factors and have a discretionary interpretation of the degree of trust using self-defined scale of trust levels. We suggested three affecting factors for trust evaluation in Section 4.4 as the credential certifier, the depth of an assertion path, and the number of assertion paths for the claimed attributes. Accordingly, the *TAP.TL* defines the target *Attribute(s)* under evaluation, the predicates for evaluation based on the deciding factors (*Factors*), and the trust level (*TrustLevel*) 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. The factor evaluation predicates are specified around the above mentioned




```
TAP_TLPolicySet = TAP_TLPolicy, {TAP_TLPolicy} ;
TAP_TLPolicy    = Attribute, {Attribute}, (CombinationOP,
                  Factors, {Factors}), TrustLevel ;
Factors         = ("Certifier" | "DlgtDepth" | "Recommenders"),
                  ComparisonOP, TargetValue ;
Attribute       = Aname, [Avalue] ;

CombinationOP   = "AND" | "OR" | "NOT" ;
ComparisonOP    = "="|"!="|">"|">="|"<"|"<=" ;
TrustLevel      = ? arbitrary string ? ;
TargetValue     = ? arbitrary string ? ;
Aname           = ? arbitrary string ? ;
Avalue          = ? arbitrary string ? ;
```

(a) ROA-TAP.TL Specification




```
TAP_TDPolicySet = TAP_TDPolicy, {TAP_TDPolicy} ;
TAP_TDPolicy    = Attribute, {Attribute},
                  ComparisonOP, Threshold ;

Attribute       = Aname, Avalue ;
ComparisonOP    = "="|"!="|">"|">="|"<"|"<=" ;
Threshold       = ? arbitrary string ? ;
Aname           = ? arbitrary string ? ;
Avalue          = ? arbitrary string ? ;
```

(b) ROA-TAP.TD Specification

Figure 5.2: ROA Policy Set Specification - Trust Management

three factors: the *Certifier*, the *DlgtDepth* and the *Recommenders*. Given an assertion path $ap[attrs] := cred_1 cred_2 \ldots cred_n$, the *Certifier* can be captured by $cred_1.certifier$; the *DlgtDepth* of the assertion path is derived by the function $depth(ap[attrs])$; and the *Recommenders* of the attributes can be captured through the function $num\_aps(attrs)$ as defined in Definition 2. Each factor is evaluated separately against the *TargetValue* according to the specified *ComparisonOP*. A boolean value is derived for each factor predicate, and a *CombinationOP* is then specified to combine the results of factor evaluations. A *TrustLevel* is finally assigned if a *true* value is derived from all factor evaluations. Figure 5.2(a) shows

the specification of *TAP.TL*.

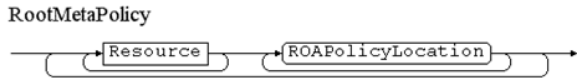
TAP.TD policy, on the other hand, specifies the threshold of the trust level for the claimed attributes to be trusted. *TAP.TD* is specified as one or more *Attributes* to be evaluated including the *Threshold* and the *ComparisonOP*. Since the trust level is defined as a partial order, we assume the default *ComparisonOP* is "$\geq$". 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. Figure 5.2(b) shows the specification of *TAP.TD*.

5.1.2 Root Meta Policy Set (RMPS)

In general, *ROA* policies are specified independently from the data resource and can be deployed in an originator's local domain for easy maintenance and update. *RMPS* is especially designed to declare the ownership of an originator's resource and associate the *ROA* policies with the *Resource* so that the policy enforcement system is able to locate and enforce *ROA* policies on behalf of the originator when data is disseminated within the collaboration community. In *RMPS*, the *Resource* is represented by a URI conforming to RFC2396 [98], and the originator is identified by the *OriginatorId* element in the format of an X.500 distinguished name (DN) [62]. The originator's *ROA* policies are referenced through the *ROAPolicyLocation* URI. Figure 5.3(a) illustrates the specification of *RMPS*.

5.1.3 Credential Policy (CREDPolicy)

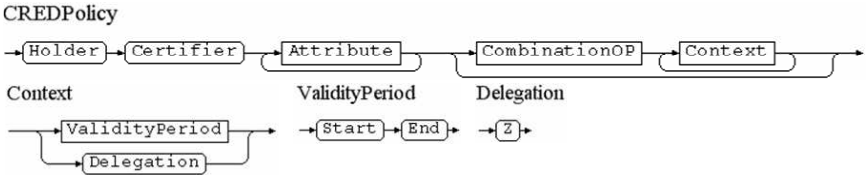
A *CREDPolicy* specifies the validation constraints for a credential. Besides the elements of *Holder*, *Certifier*, and asserted/delegated *Attributes*, each *CREDPolicy* defines one or more *Context* constraint predicates. And a logical combination operator *CombinationOP* is specified to aggregate 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




```
RootMetaPolicy      = ((Resource, {Resource}), (ROAPolicyLocation,
                      {ROAPolicyLocation})), {(Resource, {Resource}),
                      (ROAPolicyLocation, {ROAPolicyLocation})} ;
Resource            = (OriginatorId, {OriginatorId}), ResourceId ;

OriginatorId        = ? X.500 RFC2253 DN ? ;
ResourceId          = ? RFC2396 URI ? ;
ROAPolicyLocation   = ? RFC2396 URI ? ;
```

(a) RMPS Specification




```
CREDPolicy      = Holder, Certifier, Attribute, {Attribute},
                  [{CombinationOP, Context, {Context}}] ;
Context         = ValidityPeriod | Delegation ;
ValidityPeriod  = Start, End ;
Delegation      = Z ;

Holder          = ? X.500 RFC2253 DN ? ;
Certifier       = ? X.500 RFC2253 DN ? ;
Attribute       = Aname, [Avalue] ;
CombinationOP    = "AND" | "OR" | "NOT" ;
Z               = ? arbitrary natural number ? ;
```

(b) CREDPolicy Specification

Figure 5.3: Root Meta Policy and Credential Policy Specification

terms of the *Delegation* constraint, a natural number Z is defined to restrict the depth of delegation. A credential is valid when a *true* value is derived from all *Context* constraints. Figure 5.3(b) illustrates the specification of *CREDPolicy*.

As a summary, Table 5.2 elaborates all policy components and their design purposes in our RAMARS policy framework.

5.2 Policy Framework Realization Using XACML

The OASIS standard for XACML [84, 85] has been well adopted as a general policy language as well as an access decision language used to protect resources. In supporting RBAC, OASIS has recommended a specification for RBAC policies [83] ("OASIS RB-XACML" for simplicity). Inside OASIS RB-XACML, roles are specified in *Role PolicySet (RPS)*, permissions assigned to roles are defined in *Permission PolicySet (PPS)*, and user-role assignment are defined in *Role Assignment PolicySet (RAPS)* where users are as-

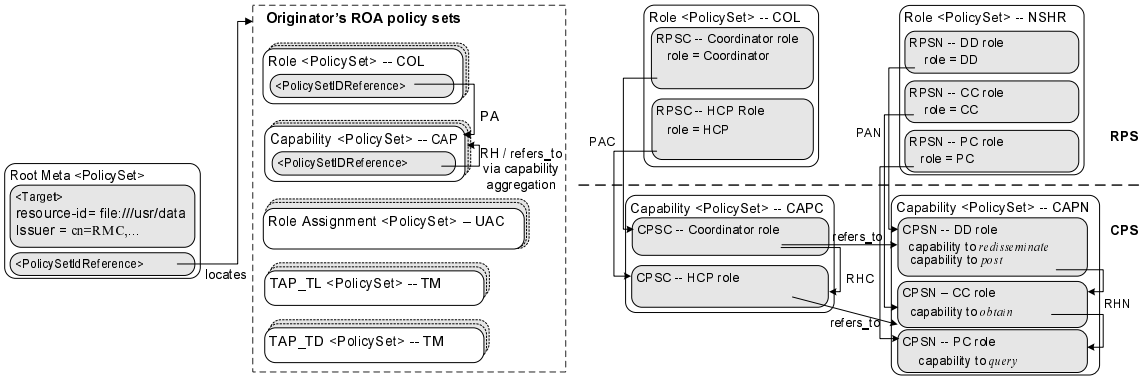
Table 5.2: Policy Framework

Component	Policy Element	Purpose
ROA Policy Set	Role Policy Set (RPS)	Declares roles and associates roles to assigned capabilities.
	Capability Policy Set (CPS)	Declares the capabilities assigned to a role; and realizes the role hierarchy and role mapping through capability aggregation.
	Role Assignment Policy Set (RAPS)	Defines rules for user-to-role assignments based on a user's attributes.
	Trust Level Assessment Policy (TAP.TL)	Defines rules to determine the trust level of user's attributes given certain supportive credentials within an assertion path.
	Trust Decision Assessment Policy (TAP.TD)	Defines rules to determine the trustworthiness of a user-attributes entitlement given the trust level achieved by the supportive credentials.
RMPS		Root policy to declare the ownership of the resource and locate an originator's ROA policies.
CREDPolicy		Credential policies defined by a credential certifier to specify context validity rules for a credential.

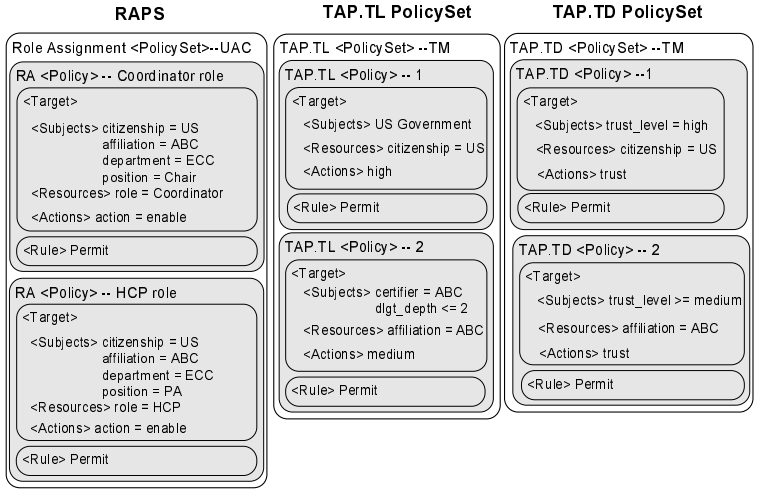
signed to the role by their user IDs. With XACML 3.0, the delegation chain is derived and validated through a process of policy reduction back to the original delegator's policy. Integrating these features and accommodating the standard syntax, we design a set of XACML-based policies as an implementation of our proposed RAMARS policy framework. Figure 5.4 illustrates the policy examples based on the scenario in Section 4.3.

Figure 5.4(a) shows an overview of the policy structure for *RMPS* and *ROA*. In *RMPS*, the *Resources* element specifies the resource in the RFC2396 URI format such as "*file:///usr/data*". The *Resources* element also associates with an *Issuer* attribute to indicate the originator of the resource in an X.500 DN format such as "*cn=RMC,...*". Inside *RMPS*, an element of *PolicySetIdReference* is included referring to the location of *RMC*'s *ROA* policies. Using LDAP as an example, the reference may be specified as "*ldap://rmc.com:389/o=RMC*".

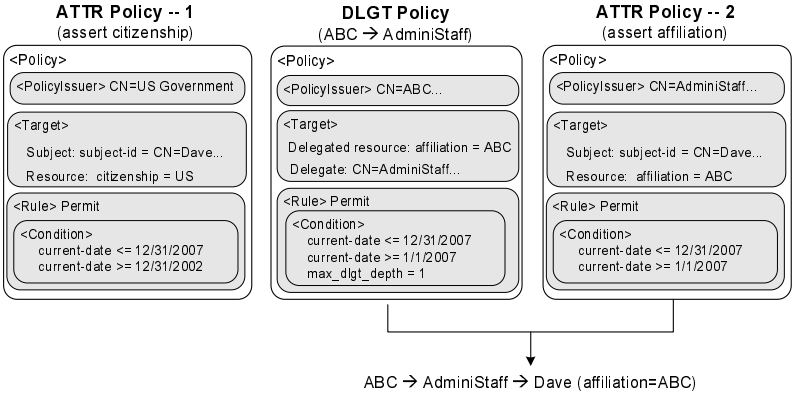
RMC's ROA policy set is composed of *RPS*, *CPS*, *RAPS*, *TAP.TL* and *TAP.TD*. *RPS* defines two collaborator roles in *RMC*'s collaborative sharing domain as *Coordinator* role and *HCP* role. Meanwhile, three normative sharing roles of *DD*, *CC* and *PC* are defined



(a) RMPS-ROA Overview (b) RPS-CPS Details



(c) RAPS-TAP Details



(d) CREDPoicy Examples

Figure 5.4: XACML Policy Examples

in *RPS* as well. The roles are defined as *Subjects* elements, and the capabilities assigned to roles are referenced by *PolicySetIdReference* elements.

CPS policy specifies the capabilities associated with these roles. The role hierarchy and role mapping are achieved by policy references. In particular, in order to represent the *Coordinator* role is senior to the *HCP* role, the *PolicyId* in the *CPS* for *HCP* role is referenced by the *CPS* for *Coordinator* role through the *PolicySetIdReference* element, so that all capabilities assigned to *HCP* role (*CPSC - HCP role*) could be loaded as part of the capability policy for *Coordinator* role during evaluation. 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 5.4(b) shows the detailed definitions in the policy sets.

In terms of role assignment, *RAPS* specifies the required attributes for *Coordinator* role and *HCP* role. Both roles require the following attributes: citizenship=US AND affiliation=ABC AND department=ECC. These required attributes are expressed as *Subjects* conditions in the policy. In addition, the chair of ECC (*position=Chair*) is assigned to the *Coordinator* role, while the normal physician assistants (*position=PA*) could be assigned to the *HCP* role.

In order to evaluate the trustworthiness of user attributes, *RMC* also defines the trust assessment policies. We assume *RMC* defines three levels of trust: *low*, *medium* and *high*, where $low \preceq medium \preceq high$. *TAP.TL PolicySet* realizes the rules on how these trust levels can be assigned to user attributes. For example, *Policy 1* of *TAP.TL* in Figure 5.4(c) specifies that when the (citizenship,US) attribute is asserted by "*CN=US Government*", a "*high*" level is assigned. *Policy 2* specifies that when the (affiliation,ABC) attribute is asserted/delegated by "*CN=ABC*", and the maximum delegation depth $dlgt\_depth \leq 2$, then a "*medium*" level of trust is assigned. Inside the policy, the trust factors are specified in the *Subjects* element, the targeted attributes are specified in the *Resources* element, and the assigned trust level is specified in the *Actions* element. The effect of the *Rule* is always specified as *Permit*, so that no negative rules are defined in our framework. In *TAP.TD*

PolicySet, *Policy 1* specifies that the (citizenship,US) attribute is trusted when its achieved trust level is *equal to high*, where *Policy 2* specifies that the (affiliation,ABC) attribute is trusted when its achieved trust level is *larger or equal to medium*. Inside *TAP.TD* policy, the trust level threshold is specified in the *Subjects* element, the attributes are specified in the *Resources* element, and "*trust*" is specified in the *Actions* element. We omit the trust assessment policies for other attributes as they are specified exactly in the same way.

According to XACML 3.0 with delegation being considered, all trusted policies do not specify the *PolicyIssuer* by default, while untrusted policies need to contain such element to check whether the policy issuer is authorized to define the policy or not. In our XACML policy implementation, all originator defined policies are considered as trusted policies that do not have the *PolicyIssuer* element, while the credential policies (*CREDPolicy*) defined by credential certifiers must have such elements to indicate the credential issuer, allowing to further evaluate the validity of the credentials. We define two separate policies as attribute credential policy (*ATTR Policy*) and delegation credential policy (*DLGT Policy*) for attribute credentials and delegation credentials, respectively. Figure 5.4(d) shows three examples in both types. In particular, *ATTR Policy 1* specifies an example of passport attribute credential/policy that "*CN=US government*" as the policy issuer asserts *Dave*'s (citizenship,US) attribute. The credential is valid within the validity period between *12/31/2002* and *12/31/2007*. The credential issuer is specified in the *PolicyIssuer* element. The holder of the credential is specified in the *Subjects* element, and the asserted attribute(s) are specified in the *Resources* element. The context constraints are specified as *Condition* within the *Rule* in order for the credential to be validated. The *DLGT Policy* specifies that "*CN=ABC*" delegates to "*CN=AdminiStaff*" the right over (affiliation,ABC) attribute within the period between *1/1/2007* and *12/31/2007*. Besides, *ABC* also specifies the *max_dlgt_depth=1* as the delegation constraint to restrict further delegations by *AdminiStaff*. The delegated attributes are specified with *Delegated Resource* category, and the delegatee is specified in *Delegate* element. *ATTR Policy 2* specifies another credential issued by *AdminiStaff* to as-

sert *Dave*'s affiliation attribute. With these two credentials, an assertion path could be established extending from *ABC* to *Dave* via *AdminiStaff* for the attribute of (affiliation,ABC). Given these three credentials, consider *RMC*'s *TAP* policies again. From the *TAP.TL PolicySet* example, we could derive that *Dave*'s (citizenship,US) attribute asserted by *US government* is assigned to a *high* level of trust, and his (affiliation,ABC) attributed asserted through the assertion path is assigned to a *medium* level of trust. From the *TAP.TD PolicySet*, we could further derive that both attributes are trusted since they meet the threshold trust levels of *high* and *medium*, respectively. Therefore, both attributes can be promoted for further role assignment evaluation.

5.3 Policy Evaluation

Figure 5.5 and Figure 5.6 depict the overall policy evaluation procedures and those are implemented in XACML by request context generation and policy evaluation. In a typical XACML evaluation environment setup, a Policy Enforcement Point (PEP) forms an access decision request. Upon receiving the request, a Policy Decision Point (PDP) retrieves relative policies from a Policy Administration Point (PAP), and evaluates the request against the retrieved policies. A response with an access *Decision* element of value *Permit*, *Deny*, *Indeterminate* or *NotApplicable* is made and returned to the PEP for further authorization enforcement. Accordingly, we introduce a *Context Handler* as a subcomponent of the PDP to conduct a series of XACML request generation and decision processing operations for a single access request sent by the PEP. We further introduce three sub-PDP components – *TM PDP*, *Role PDP* and *AuthZ PDP* – to take the responsibilities of different evaluation stages. We now focus on the evaluation process, and leave the detailed explanation of system components to Chapter 6. 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 Figure 4.3: passport, driver's license, employment letter, outsourcing letter, PA position on-duty authorization and PA position entitlement letter.

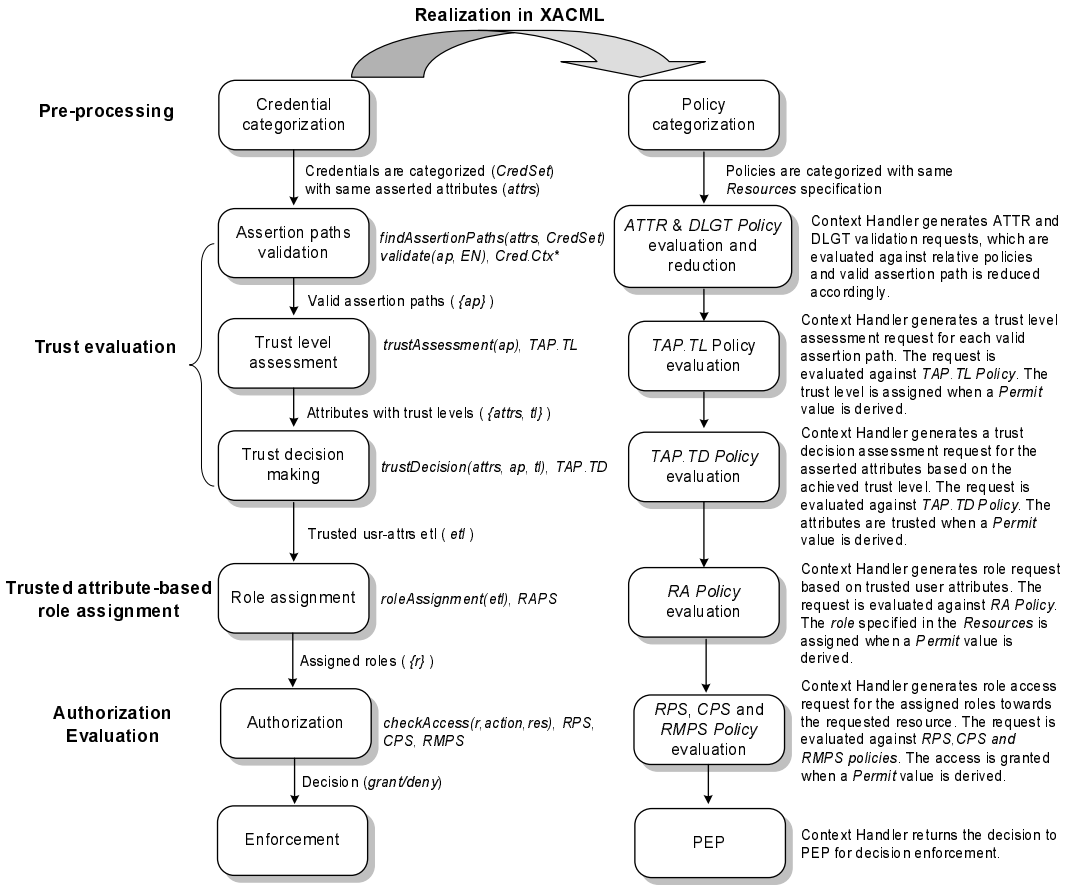


Figure 5.5: Evaluation Realization in XACML

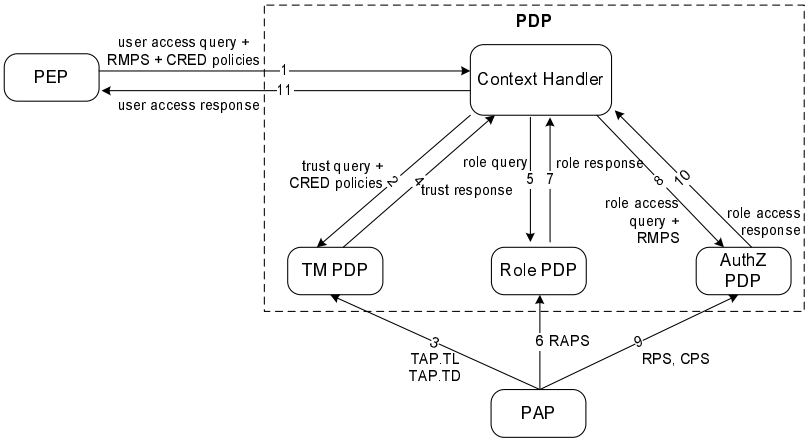


Figure 5.6: XACML Evaluation Flow

1. **Pre-processing:** The evaluation process is triggered when PEP forwards *Dave*'s file access request with supportive credentials to PDP. In our system settings, the root policy *RMPS* is associated with the resource, from which PDP engine could locate and retrieve all *ROA* policies from the originator's PAP policy store. A preprocessing operation is conducted to categorize the user credentials according to their asserted attributes. In XACML evaluation environment, the *Context Handler* parses and groups the *CRED Policies* based on same *Resource* and *Delegated resource* attributes. In our example, Dave's six credentials are categorized into four groups to assert the attributes of *citizenship*, *affiliation*, *department* and *position*. The *Context Handler* then generates a series of trust evaluation queries for the *TM PDP* to evaluate.

2. **Trust evaluation:** The first step of trust evaluation is to derive and validate the assertion paths for each attribute under evaluation. This is realized in XACML through a process of policy reduction. In particular, the *Context Handler* first generates an *attribute validation request* for *TM PDP* to evaluate. As shown in Figure 5.7, the attribute validation request queries whether *Dave* (*CN=Dave...*) is authorized to possess the (affiliation,ABC) attribute on *current date* (*6/1/2007*). The *TM PDP* locates the employment credential issued by AdminiStaff (*ATTR Policy 2*) as the *Target* matches the request, and evaluates the validity period constraint against the environmental parameter *current date*. As the *current date* meets the *Condition* in the Rule (*1/1/2007*$\leq$*current-date*$\leq$*12/31/2007*), a *Permit* decision is derived from the evaluation. Since *ATTR Policy 2* is issued by *AdminiStaff*, the *Permit* decision means that *AdminiStaff* asserts *Dave*'s *affiliation* attribute. However, the system has to further make sure that *AdminiStaff* is an authorized/trusted certifier to assert the attribute. The *Context Handler* then generates a *delegation validation request* to query whether the attribute *affiliation=ABC* is delegated to *CN=AdminiStaff* on *current date* (*6/1/2007*) as delegation *depth=1*. The *TM PDP* locates ABC's outsourcing

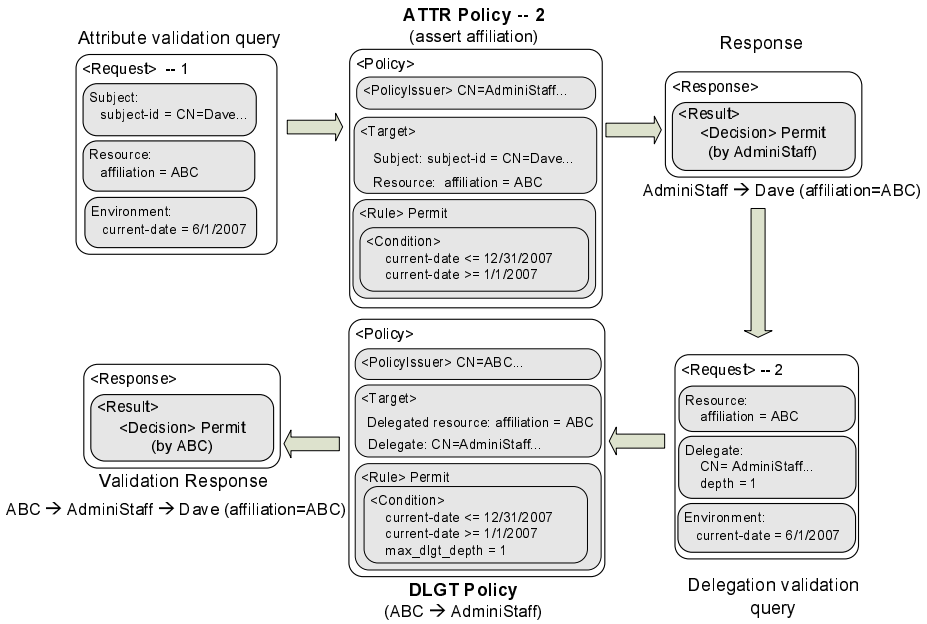


Figure 5.7: Assertion path validation through policy reduction

credential (*DLGT Policy*) and a *Permit* decision indicates that *AdminiStaff* is a legitimate entity to assert (affiliation,ABC) attribute on behalf of *ABC*. Since there are no other credentials available regarding the (affiliation,ABC) attribute, the policy reduction stops here and a valid assertion path is derived as *ABC→AdminiStaff→Dave*, with *Certifier=ABC* and *DlgtDepth=2*. Figure 5.7 shows the whole process of policy reduction.

The next step of evaluation is to determine the trust level of the asserted attributes given valid assertion paths. In particular, the *Context Handler* generates a *trust level assessment request* for *TM PDP* to evaluate whether a particular trust level can be assigned to an attribute given the assertion path. *TM PDP* retrieves *TAP.TL* from PAP and evaluates the request. And when a *Permit* decision is derived, the trust level is assigned to the asserted attribute. According to *Policy 2* in *TAP.TL* (Figure 5.4(c)), the assertion path for the (affiliation,ABC) attribute with *Certifier=ABC* and *DlgtDepth=2* is assigned to a trust level of *medium*.

Finally, the system decides on the trustworthiness of the attribute given the achieved

trust level. For example, the *Context Handler* generates a *trust decision request* to query whether the attribute (affiliation,ABC) with a *medium* level of trust can be trusted or not. *TM PDP* evaluates the request against the *TAP.TD*. A *Permit* decision indicates that the attribute is trusted. According to the example *TAP.TD* policy in Figure 5.4(c), the attribute meets the minimum required trust level ($\succeq$ *medium*) and thus is trusted for the role assignment. A *trust response* with *Permit* decision is then sent back from *TM PDP* to the *Context Handler* for further evaluation.

3. **Role assignment evaluation:** After the trust evaluation, all trusted user attributes are promoted. For each role in the system, the *Context Hander* generates a *role query* for *Role PDP* to evaluate. For example, if *Dave*'s attributes of *citizenship*, *affiliation*, *membership* and *role* are all trusted in the trust evaluation, a role query is generated to query whether *Dave* is authorized to *enable* an *HCP* role given his trusted attributes. The *RAPS* policy is retrieved from PAP for evaluation. According to the example *RAPS* policy in Figure 5.4, a *Permit* decision is derived, and the *HCP* role is eventually assigned to *Dave*.

4. **Authorization evaluation:** Finally, the Context Handler generates a *role access query* for *AuthZ PDP* to evaluate whether *HCP* role is authorized to conduct the "*obtain*" action on the file resource (*file:///usr/data*). The request is evaluated against *RMPS*, *RPS* and *CPS* policies. As stated in the example policies in Figure 5.4, *HCP* role is mapped to *CC* role and *CC* role is allowed to *obtain* the file. Hence *HCP* role is allowed to *obtain* the file. A *Permit* decision is sent back to the *Context Handler* as the *role access response*. The final decision is forwarded from *Context Handler* back to PEP for decision enforcement, where *Dave*'s access to the medical files is eventually granted.

Summary: In this Chapter, we introduced the formal policy specification for RAMARS framework in EBNF format. The RAMARS policies were implemented using standard

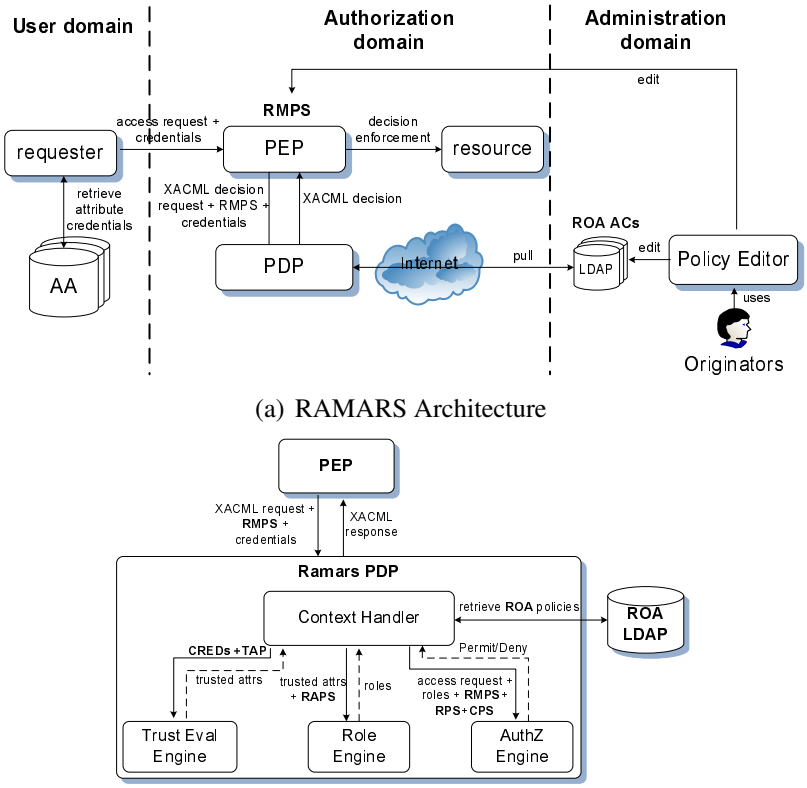
XACML policy language and we discussed the policy evaluation procedure for the policy engine to evaluate the policies and make an authorization decision. In Chapter 6, we introduce necessary system components to carry out the policy evaluation functionalities. We further demonstrate the feasibility of our proposed RAMARS authorization system by implementing prototype systems in supporting secure information sharing in both P2P networking and Grids computing environments.

CHAPTER 6: SYSTEM DESIGN AND PROTOTYPE IMPLEMENTATION

6.1 RAMARS Authorization System Architecture

We have designed and implemented a prototype system to demonstrate how the proposed RAMARS access management framework and policy specification can be realized as comprehensive authorization services within the context of collaborative sharing applications.

RAMARS system is designed to be deployed in distributed collaborative environments without assuming any centralized policy store. The architecture of RAMARS system can be segregated into three domains as shown in Figure 6.1(a). In the administration domain, resource originators edit and maintain their ROA authorization policies by using the Policy Editor, which is a facility toolkit for originators to create and modify ROA policies. These policies are stored in the originator's perspective policy store (i.e., LDAP directory server) serving as the Policy Administration Point (PAP) for RAMARS authorizations. Once ROA policies are created, they are expected to be automatically enforced by the RAMARS authorization system without further interception by the originators in the collaborative sharing service systems. In the authorization domain, the RMPS policy is always associated with the data resource for RAMARS PDP to locate the originator's ROA policies. Upon receiving an access request and supportive credentials coming from the user domain, the Policy Enforcement Point (PEP) invokes RAMARS PDP with a formulated access decision request, RMPS policy and the user's credentials. RAMARS PDP, as illustrated in Figure 6.1(b), consists of four subcomponents to carry out the necessary policy evaluation functionalities as discussed in Chapter 5: Context Handler, Trust Evaluation Engine, Role Engine and Authorization Engine. In particular, the Context Handler dynamically retrieves ROA policies from the originator's policy store based on the location references specified



(a) RAMARS Architecture

(b) RAMARS PDP and Policy Evaluation

Figure 6.1: RAMARS System Architecture

in RMPS. The requester's credentials and the originator's TAP policies are sent to Trust Evaluation Engine where trusted attributes are derived. These trusted attributes and RAPS policy are carried out by the Role Engine to determine the user's assigned collaborator roles. Given the assigned roles, the user's access request is evaluated against policies of RMPS, RPS and CPS, and the Authorization Engine makes the final access decision. This access decision is sent back to PEP as an XACML response for decision enforcement.

In the RAMARS system, the user's credentials are "pushed" by the requester to the authorization system, while the originator's ROA policies are dynamically "pulled" at run-time upon policy evaluation. Therefore, the deployment of RAMARS PDP does not rely on any static configuration with any centralized policy store, making RAMARS PDP a general purpose authorization engine to serve various collaborative sharing applications. To demonstrate the feasibility of our proposed RAMARS system, we adapt and integrate

our RAMARS system into both P2P networking and Grid computing collaborative infrastructures as *ShareEnabler* and *RamarsAuthZ service*, respectively, to support secure and authorized information sharing within collaborations.

6.2 RAMARS in P2P – ShareEnabler System

6.2.1 ShareEnabler System Architecture and Operation

ShareEnabler adopts a specific communication infrastructure from a P2P based scientific information sharing toolkit SciShare [18] developed by Lawrence Berkeley National Laboratory. In our P2P-based collaborative sharing system, each participating entity is represented by a ShareEnabler agent that executes sharing services on the participant's behalf. An 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 between a pair of peer agents. Figure 6.2 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 1 - 5 in Agent 1). 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 (step 2' - 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, while Agent 2 further checks with the originator's *ROA* policies and initiates the data transferring process if the requester is authorized to download the resource. We call this process as *data sharing*. The access control for processes of both metadata sharing and data sharing is carried out by our proposed RAMARS system.

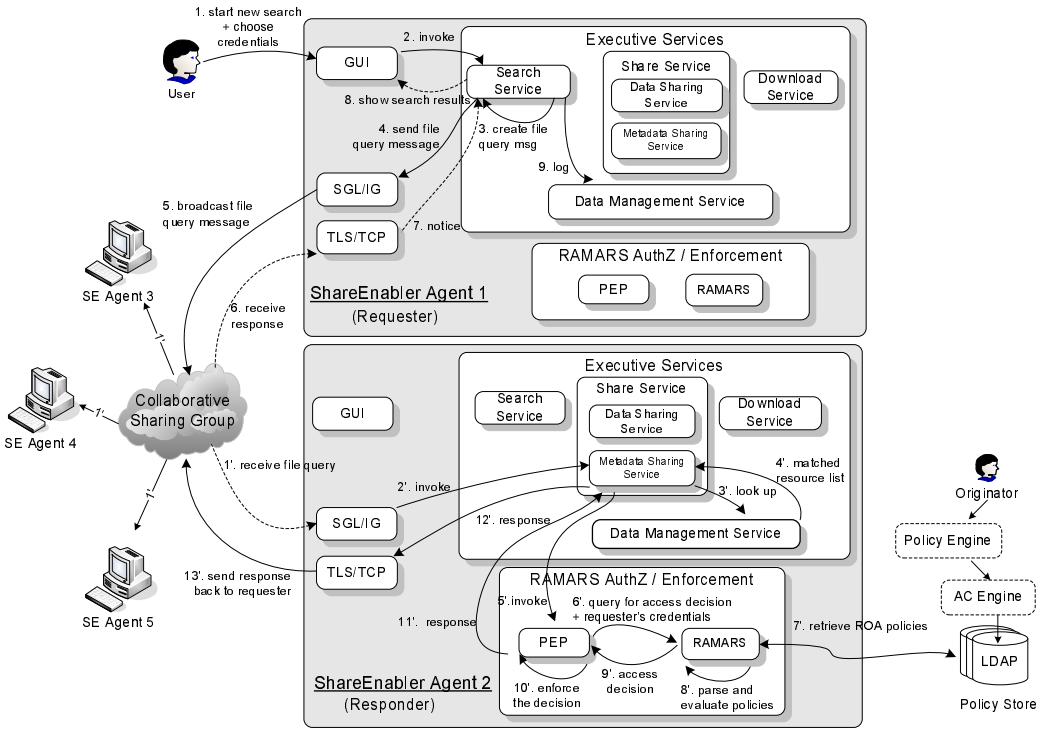


Figure 6.2: ShareEnabler System Infrastructure and Architecture

Figure 6.2 also shows the system components inside a ShareEnabler Agent and their interactions in the process of *metadata sharing* between the ShareEnabler Agent 1 as the 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)) [8, 69] 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 P2P collaborative sharing behaviors, which include Search, Download and Share Services. All these services interact with a Data Management Service serving as the background database in the agent. The RAMARS AuthZ/Enforcement service is the central component that conveys our proposed

RAMARS system for the core access and dissemination control services. As shown in Figure 6.3, the PEP is responsible for the request processing and access decision enforcement.

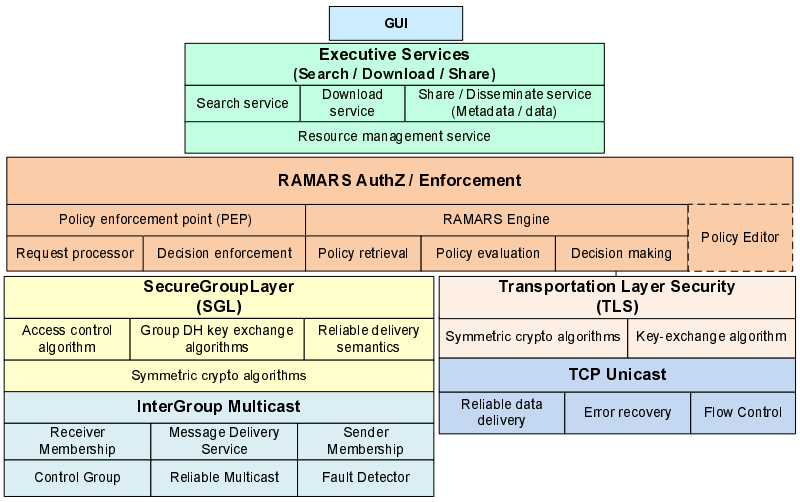


Figure 6.3: Detailed Components in a ShareEnabler Agent

The RAMARS Engine is designed for the policy retrieval and all policy related evaluations, and the detailed internal architecture of RAMARS Engine complies to the RAMARS PDP in Figure 6.1(b). The Administrative Policy Editor can also be invoked from this component for an originator to create and edit ROA policies at anytime while operating the ShareEnabler Agent. The secure and reliable multicast communication is achieved by the combination of the Secure Group Layer (SGL) [8] and the InterGroup protocol (IG) [69]. In particular, SGL supports group key-exchange and symmetric cryptography algorithms to enable group peers to establish a shared session key, authenticate each other and encrypt all communication channels. IG protocol provides reliable ordered message delivery, membership notification, group control as well as fault detector mechanisms for the multicast communications. In the category of unicast communication, similar functionalities are achieved by the Transport Layer Security (TLS) and Transmission Control Protocol (TCP) when two peers play the traditional roles of a client and a server, respectively.

In the process of metadata sharing, on the requester agent side (ShareEnabler Agent 1), a user interacts with the GUI to specify the query criteria and choose his/her credentials[1]

[1]Here we assume all users' credentials are locally stored for the demonstration purpose of authorization services. We demonstrate how our system can be extended to support credential retrievals from online credential providers in Section 6.3.

(step 1). GUI invokes the Search Service to formulate and embed the user's credentials into a query message, and broadcast to all peers in the collaborative sharing group through SGL/IG (step 2 - 5). Upon receiving responses from other peers, TLS/TCP notices the Search Service with the response messages (step 6 - 7), and these responses are parsed and then shown in the GUI (step 8). The search results are backed up through Data Management as well (step 9).

On the responder agent side (ShareEnabler Agent 2), the SGL/IG module notices the Metadata Sharing Service (step 1' - 2') upon receiving the query message. The Metadata Sharing Service separates the content request and the requester's credentials from the query, then invokes the Data Management Service to find matched resources against the content request (step 3'). For resources listed in the agent, the originator defines and stores its *ROA* policies in a localized LDAP policy repository using a facility Administrative Policy Editor tool, while the root policy *RMPS* is attached with the resource in the ShareEnabler agent. The Data Management Service returns a list of matched resources along with their associated root policies (*RMPS*) to the Metadata Sharing Service, through which the PEP is invoked for access checking and enforcement (step 4' - 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 relative *ROA* policies from the originator's LDAP directory and examine whether the requester is allowed to *query* the resource following the whole evaluation process as we discussed in Section 5.3 (step 7' - 9'). Upon receiving the access decision from the RAMARS Engine, PEP enforces the decision by removing unauthorized resources from the list, so that only authorized resources are returned to the Metadata Sharing Service (step 10' - 11'). Finally, the Metadata Sharing Service formulates the response message and sends back to the requester using the TLS/TCP protocol (step 12' - 13').

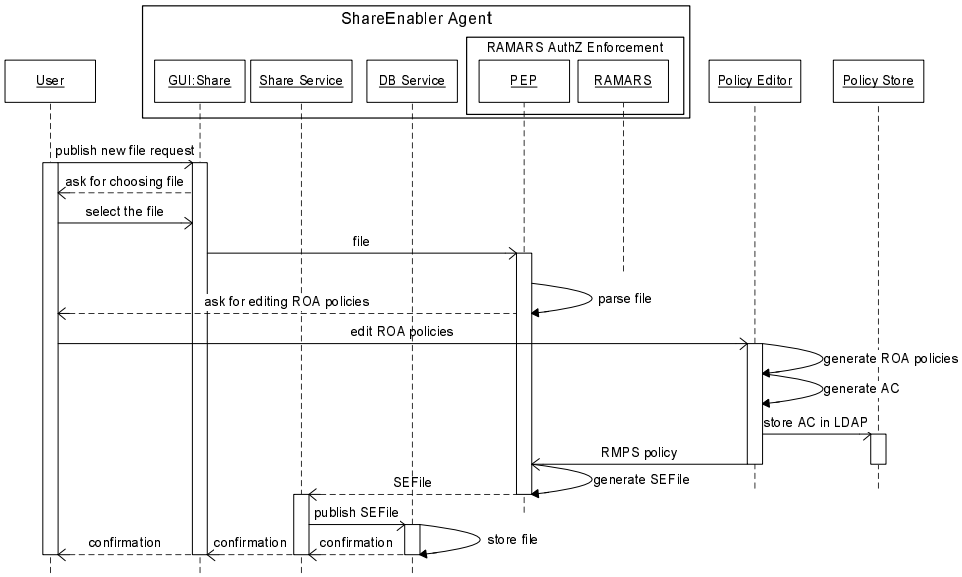
In our system, *ROA* policies are maintained in an originator's administrative domain, which are deployed separately from the major ShareEnabler application and enforcement

components. A lightweight *RMPS* policy is attached with the originator's resources in ShareEnabler so that the *ROA* policies are located and pulled at runtime when the RAMARS AuthZ/Enforcement module needs to make an authorization decision. Hence, 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 an originator's *ROA* policies. The X.509 Attribute Certificate (AC) is a basic data structure in Privilege Management Infrastructure (PMI) [64] to bind a set of attributes to its holder. With its portability and flexibility, AC is considered as an ideal container of 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 attribute certificates and store in the originator's LDAP policy repositories.

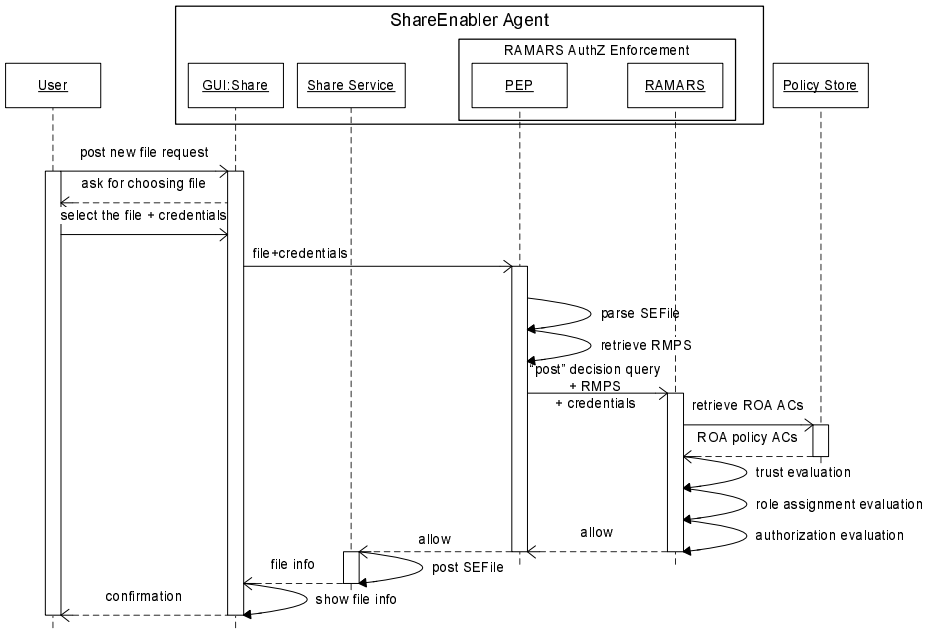
In terms of a user's credentials, X.509 public key certificate (X.509 PKC) is the major identity credential for each ShareEnabler agent to authenticate itself to other agents for secure communications within the P2P sharing community. The certificate can be either self-signed or signed by a trusted certificate authority. The self-signed certificate could be used by a new peer agent (called pseudo user) to join the community quickly. However, the X.509 PKC is not the major credential to determine a user's privileges. As illustrated in our RAMARS framework, a user's privileges are determined through the attributes he/she possesses, which are bound into X.509 ACs. 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 encode the XACML *CRED Policies* as attributes in ACs to be transferred between peer agents for authorization purposes.

6.2.2 Dissemination Control Mechanisms

The goal of access and dissemination control for ShareEnabler is to guarantee that the resource is shared within an originator's collaborative sharing domain as defined by *ROA* policies. Our system applies a distributed policy propagation and enforcement scheme



(a) Resource Publication by An Originator



(b) Resource Re-dissemination by A Collaborator

Figure 6.4: Sequence Diagrams for Dissemination Control

with decentralized, self-enforcing, and self-monitoring features at each ShareEnabler agent level. In particular, each ShareEnabler agent ensures that an originator's *ROA* policies are enforced locally by the RAMARS AuthZ/Enforcement component, so that only legitimate peers can obtain the requested file. Meanwhile, these *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 a ShareEnabler Agent to locate and enforce an originator's *ROA* policies, it is essential to make sure that *RMPS* is propagated along with the file dissemination, and its 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 an original data file with its associated *RMPS* policy. When an originator tries to publish a file in its ShareEnabler agent, the originator is prompted to interact with the Administrative Policy Editor to edit and store *ROA* policies in the originator's LDAP policy directory. Meanwhile, the PEP creates the SEFile from the original data file and *RMPS* policy. The SEFile is encrypted with a predefined secrete key and stored in the ShareEnabler agent to be shared with all other collaborators. 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. Figure 6.4(a) illustrates the sequence diagram for an originator to publish a data file in its ShareEnabler agent with *ROA* policies creation and SEFile generation.

To prevent unauthorized dissemination, when a collaborator tries to *post* a pre-obtained data file in his/her ShareEnabler agent for re-distribution purpose. The SEFile is detected and decrypted by the PEP to get the original *RMPS* policy. RAMARS Engine is prompted to retrieve the originator's *ROA* policies as indicated in *RMPS* and makes an access control 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/she is not authorized to do so. Otherwise, ShareEnabler automatically posts the resource in the SEFile format and allows it to be re-disseminated. Figure 6.4(b) illustrates the sequence diagram of this process.

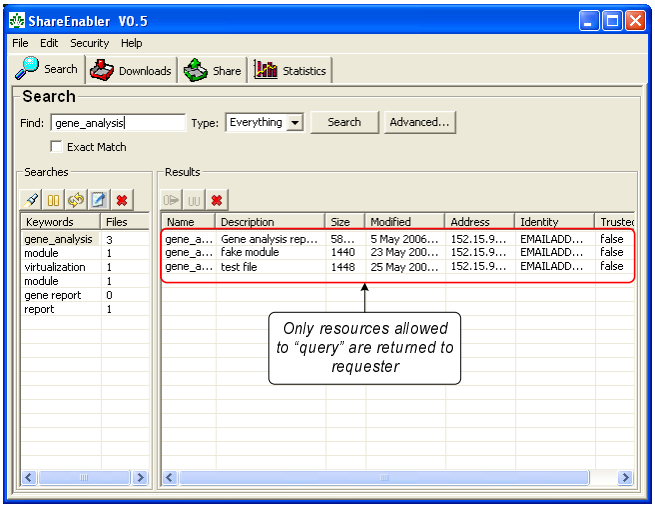
6.2.3 Implementation

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 [18] for the implementation of SGL/TLS communication protocol as well as the basic authentication mechanisms underneath. We extend SICS's XACML3.0 implementation [106] 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 [59] is used to implement major cryptographic and X.509 attribute certificate related modules. Finally, the IPlanet Directory Server serves as the back-end LDAP policy repository.

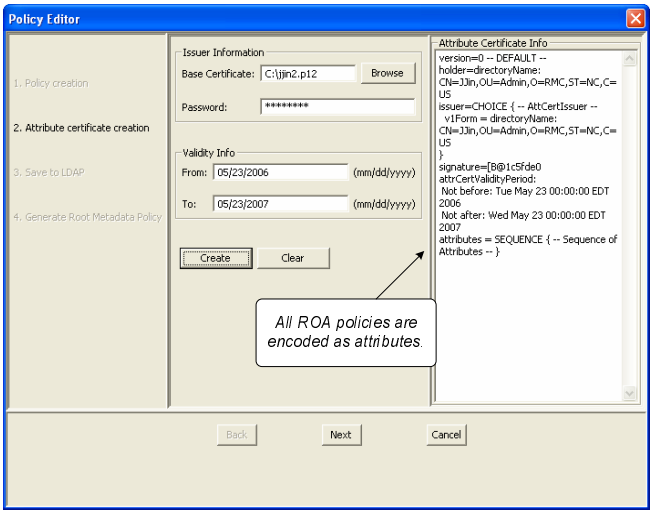
Figure 6.5(a) shows an interface when a user *Dave* searches a particular file resource. For the responding agent, only the data resources that *Dave* is authorized to *query* are returned back and shown in *Dave*'s agent. A sample authorization message on the responder side is also shown under the interface. Figure 6.5(b) shows an interface of Administrative Policy Editor for the originator *RMC* to create *ROA* policy X.509 AC. All *ROA* policies are encoded as attributes in the certificate.

6.2.4 Performance Evaluation and System Improvement

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



(a) New Search and Search Results



(b) Attribute Certificate Creation

Figure 6.5: User Interfaces of ShareEnabler Agent and Administrative Policy Editor

the authorization module scales along with the increased evaluation complexity and also analyze the overhead that RAMARS Engine has put on 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. RAMARS Engine takes an access request along with a set of supportive credentials in X.509 ACs 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 of the policy evaluation. 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 without hierarchies. 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 6.6 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 $p = 2m$ 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 512MB RAM running at 1.8GHz with Windows XP. The time consumption for each evaluation process is measured

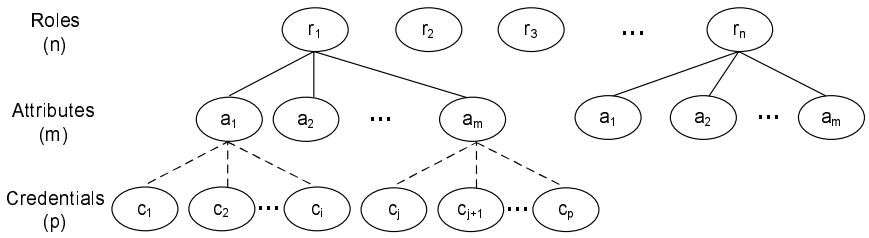


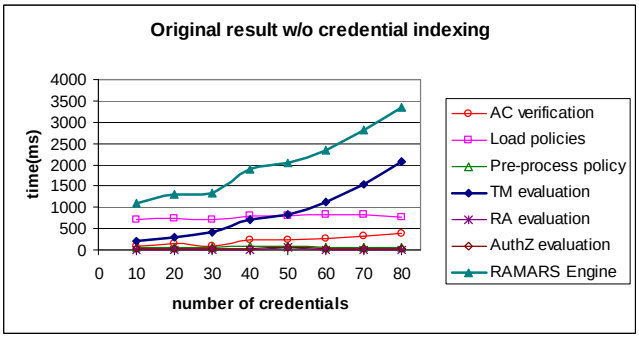
Figure 6.6: RAMARS Engine Performance Testcase Setting

in milliseconds and calculated as the average of 100 independent test runs [2].

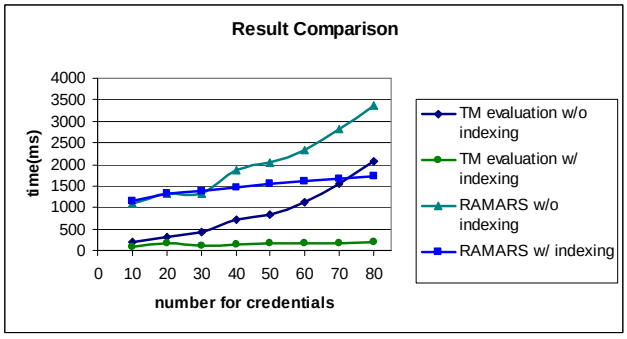
Test 1 – Credential increase and credential indexing

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 to handle 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 6.7(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

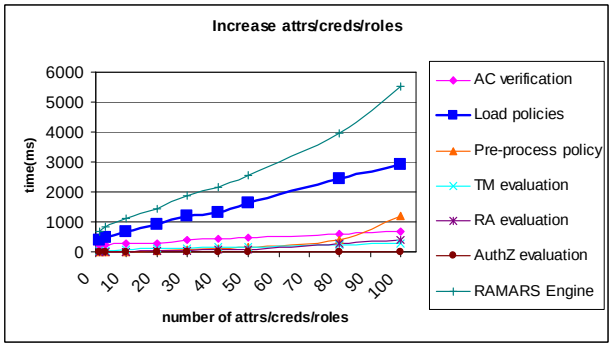
[2]For brevity, our performance analysis omitted the network-related overhead.



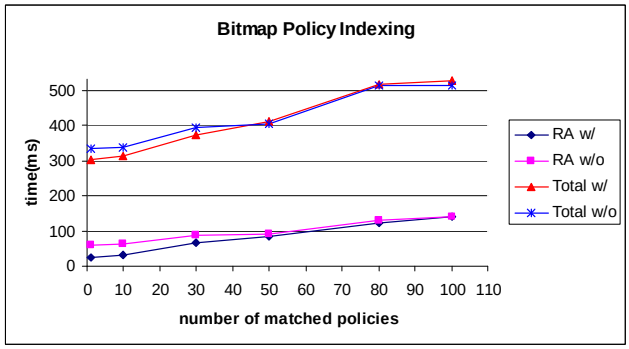
(a) Result without credential indexing



(b) Comparison with credential indexing



(c) Increasing roles and attributes



(d) Comparison with bitmap policy indexing

Figure 6.7: Performance Testing Results

result of our initial test, as shown in Figure 6.7(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 seconds to less than 2 seconds with a flatter increase trend. The time consumption of TM evaluation stays almost stable at around 0.3 seconds. This is a desirable property and has made the system more scalable to a large number of credentials.

Test 2 – Role / attribute increase and bitmap indexing

Our next experiment is conducted to examine how well the authorization module scales when the roles and required attributes increase in *ROA* policies. Figure 6.7(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 collaborator roles in its collaborative sharing domain and require 100 attributes for each role assignment. 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 [49], 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 an efficient XML parser is beyond the scope of this research.

As another effort in improving the performance of policy evaluation, we investigate bitmap-based indexing technologies [115] 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

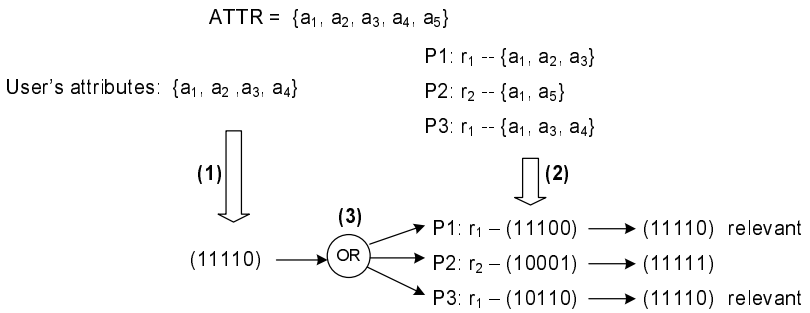


Figure 6.8: Bitmap Policy Indexing

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 policy defines the required attributes for a particular role. The role assignment evaluation then 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 to make sure 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 a bit array: 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 Figure 6.8, 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 a bit array indexed by the role: for each role assignment policy element 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 Figure 6.8, 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 to index policies and conduct 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 Figure 6.7(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 have clearly observed the potential

Table 6.1: RAMARS Overhead Analysis

Testcase (*attr,cred,role*)	Base	(10,10,10)	(20,20,20)	(30,30,30)
Total time (*seconds*)	487.5	488.0	489.0	490.8
Overhead (%)	0.00	0.10	0.30	0.68
Testcase (*attr,cred,role*)	(40,40,40)	(50,50,50)	(80,80,80)	(100,100,100)
Total time (*seconds*)	492.8	495.4	499.8	508.3
Overhead (%)	1.07	1.61	2.53	4.27

for the bitmap indexing mechanism 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 [77] 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 DØ Experiment [42, 45], 300 DØ users submitted 15,000 requests involving 2-4TB data transfer per day with an average of 130M bytes data transfer per query. We choose a sample data file of size 121,781KB 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 512MB RAM running at 2.52GHz with Windows XP. Table 6.2 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.

6.3 RAMARS in Grid Computing – RamarsAuthZ System

Grid computing is considered as another important technology aiming at secure and efficient resource sharing within virtual communities. Being proposed as a generic approach, the RAMARS framework should be flexible and extensible to support secure information sharing in various collaborative environments. Therefore, we further evaluate how RAMARS framework can be adapted for collaborations in Grid computing environments enabling the same effective access control as it has been demonstrated in P2P environments.

6.3.1 Access Control Challenges in Grid Computing

Grid data and resources by nature are diverse in their locations, types, structures, ownerships, naming conventions and access capabilities. By embracing the service-oriented approach [46, 54] and the recent Web Services technologies, resources in Grid communities are uniformly represented and shared through Grid services with well-defined interfaces for dynamic service creation, resource discovery, lifetime management, and so on [46]. Examples of such Grid data sharing services are Data Replication Service (DRS) in Globus Toolkit [29, 1], Open Grid Services Architecture - Data Access and Integration (OGSA-DAI) [13] and Storage Resource Broker (SRB) [97]. However, the facility provided to Grid clients requires more advanced Grid access control mechanisms to accommodate unique challenges ranging from from the authorization model to the system architecture and deployment.

Firstly, Grid systems are usually composed of a number of dynamic and autonomous domains involving a large number of distributed users. An effective and manageable authorization scheme is necessary for the resource owners to control the access and sharing of their resources. Attribute-based access control (ABAC), which makes decisions relying on attributes of requesters, resources, and environment, has been widely adopted as a scalable and flexible authorization solution for highly distributed Grid environments [71]. In an attribute-based authorization system, the entity that manages user attributes is referred to as an Identity Provider (IdP). A user's attributes are normally collected by multiple IdPs in Grids. For example, a user is associated with a "home institution" which typically manages his employment status and affiliation attributes, while another IdP is associated with a Grid Virtual Organization (VO) that maintains attributes such as membership and role information within the Grid. The authorization system that supports ABAC in Grids then needs to be seamlessly integrated with all related IdPs and delivers user attributes in a secure and trusted manner.

Secondly, from the system architecture and deployment perspective, there are a number of dimensions to be considered for an attribute-based authorization system. In terms of the attribute collection process, the "push" strategy requires the clients to obtain and push their attributes to the Grid service at the initial request. The "pull" strategy, on the other hand, does not require the clients to submit any attribute. Instead, it is the responsibility for the authorization system to acquire attributes from the client's IdPs. While the clients have more options to select the attributes being released for authorization in the "push" mode, the "pull" mode simplifies the overall interception by the clients. It is impossible to determine which mode is more suitable for dynamic Grid environments. However, it is highly desirable for the authorization system be flexible enough to cope with both options. In terms of system deployment, the reliance on statically configured modules to render an authorization decision such as policy and attribute management should be minimized, as the authorization system may serve the authorization functions for various Grid services

running within the infrastructure.

Finally, the data resources being shared through the Grid data sharing service normally belong to different institutions. These institutions, as the owners of the data resources, should directly participate in defining authorization policies for their data sets, and their authorizations need to be efficiently conveyed and effectively enforced within the Grid data sharing service. Meanwhile, the access and invocation of Grid data sharing service itself apparently needs to be well protected to accommodate the security requirements of its service provider. Therefore, it is required for authorization systems to be flexible enough to synthesize both service-level and data-level controls accommodating security policies from different stakeholders such as the data resource providers and the service providers.

In accommodating the above-mentioned requirements, we design and extend our RAMARS framework as a flexible policy-driven authorization system, called RamarsAuthZ, for data sharing and management services in Grids. We seamlessly incorporate the trust-aware role-based authorization functions of RAMARS authorization infrastructure in the Grid Data Replication Service (DRS) to provide a fine-grained originator control for both the Grid DRS service and the data being shared through the service. RamarsAuthZ does not rely on centrally configured policy stores to make authorization decisions. It is interoperable with several state-of-the-art Grid technologies including Globus Toolkit version 4 (GT4) [12], Globus DRS service [29, 1], Shibboleth [25] and GridShib [55].

6.3.2 Leveraged Technologies

In this Section, we give a brief overview of the leveraged technologies in our system to fully integrate with the existing Grid computing infrastructure.

Globus Toolkit: The open source Globus Toolkit [12] is the core Grid infrastructure that serves as a container hosting a number of Grid services for running Grid jobs including resource monitoring, discovery, and management. Inside Globus Toolkit version 4 (GT4), the Grid Security Infrastructure (GSI) [113] is implemented as the de-facto solution to

provide the fundamental security services such as authentication and message protection for Grid environments. In particular, X.509 Proxy Certificate [110] is utilized in GSI to allow a Grid user to periodically delegate his/her identity and privileges to another entity so that the bearer is able to authenticate and establish secure connections with other parties on the Grid user's behalf. In our system, we explore the proxy certificate to convey attributes and other necessary information from Grid client to RamarsAuthZ authorization system so that the system can be dynamically configured to support both the "push" and "pull" modes.

GT4 server-side authorization framework encapsulates a set of built-in Policy Decision Point (PDP) modules. The default authorization PDP in GT4 evaluates an access control list (ACL) type of policy located in a *gridmap* file, which specifies mappings of a user's global identity (called distinguished name, DN) to a local account. Users are authorized to use the resources when their DNs appear in such list and the privileges are determined by the associated local account. This authorization approach is primitive and does not scale to a large number of Grid users. More recently, the Global Grid Forum (GGF) has proposed a SAML AuthZ specification [112] using SAML [78] as a standard message format for requesting and expressing authorization assertions so that external authorization systems can remotely make authorization decisions and respond authorization queries. With this approach, it is required for the external authorization system to render an access decision based on the information conveyed by a SAML request with least dependency on static configurations. Our RamarsAuthZ authorization system fully supports the SAML-based method to make authorization decisions for the enhanced DRS.

Globus Data Replication Service (DRS): GT4 provides a number of components to enable collaborative data sharing concerning with the discovery, access and dissemination [52]. The Replica Location Service (RLS) is responsible for the data registration and enables the discovery of data resources. Inside an RLS, a unique identifier called *logical name* is created for each registered data item. A mapping of *logical name–physical location* is

maintained for the data item and its replicas. For instance, "*GeneSequence–gsiftp://abc.com/var/gseq.tar*" states an entry in RLS. By querying the logical name "*GeneSequence*," a user could locate his/her desired data item at "*gsiftp://abc.com/var/gseq.tar*." When a data resource is discovered, the Reliable File Transfer service (RFT) is in charge of data replication to the target location. Globus Data Replication Service (DRS) [29, 1] is developed as a higher-level data management service incorporating the functionalities of both RLS and RFT services. In particular, upon receiving a user's file replication request, the DRS begins by querying RLS to discover the existence of the desired files. Then the DRS invokes RFT to replicate the files. When the file transfers are completed, the new replicas are finally registered with RLS so that they can be further discovered and disseminated. In terms of authorization, the DRS service alone can be configured with GT4 built-in authorization mechanisms. Therefore, only authorized Grid users can invoke the service and exercise the data replication functions. However, such configuration cannot further protect the actual data resources that are replicated through DRS. Our system largely enhances the authorization in DRS by enforcing both the service-level and data-level controls based on different stakeholders' policies.

Shibboleth and GridShib: Shibboleth is an Internet2 middleware initiative project aiming at providing cross-domain single sign-on and attribute-based authorization based on SAML [25]. Shibboleth leverages the identity federation between educational organizations so that a user can authenticate on his/her own campus and access to remote resources where his attributes are passed to the resource providers for authorization. There are two major separately deployed components in Shibboleth infrastructure: the Identity Provider (Shib-IdP) and Service Provider (Shib-SP). A Shib-IdP manages a user's identities and asserts his/her attributes. A Shib-SP, on the other hand, resides at the resource side to request the remote user's attributes from his/her Shib-IdP and enforces attribute-based access control. All participating Shib-IdPs and Shib-SPs are managed in a Shibboleth federation, where their trust relationships are bridged through the centralized Shibboleth federation

certificate authority.

GridShib [55] project is initiated to integrate Shibboleth infrastructure with Grid technology. A set of tools has been developed to facilitate Grid resources to fetch attributes from Shibboleht IdP. However, very limited attribute-based authorization functionalities can be achieved by GridShib. Most of early adopters [37, 56] of GridShib only implement the "pull" mode of attribute acquisition, relying on pre-configured single source of Shib-IdP. This approach lacks flexibility and is insufficient to support the attribute-based access control based on multiple IdP sources. In addition, as Shibboleth federation is separated from the common trust base established in Grid VO, the attributes issued by Shibboleth IdP are not necessarily trusted by Grid resources. The trust management issue is not well addressed either in current available systems. In our system, we explore a feasible and integrated solution to these issues.

6.3.3 Integrated RamarsAuthZ System

Recall in our original RAMARS system, a user "pushes" an access request along with his/her credentials to the authorization system, while RAMARS PDP dynamically locates and pulls an originator's ROA policies upon policy evaluation. The deployment of RAMARS PDP does not need to be configured with any centralized policy store. In our RamarsAuthZ system, we enhance the existing system by supporting the hybrid "push" and "pull" modes for attribute acquisition. And we also extend the RAMARS PDP as a Grid Authorization Service exchanging standard SAML authorization messages with Grid services.

Figure 6.9 illustrates the integrated system architecture for RAMARS, Globus Toolkit, Shibboleth and Globus DRS data sharing service. Inside a Globus Toolkit Container, a RAMARS Adaptor is introduced for the DRS service as an authorization plugin to communicate with the RamarsAuthZ service which is essentially based on RAMARS PDP. The RamarsAuthZ service can be deployed and called out by the RAMARS Adaptor through two mechanisms: a localized function call by API or a remote service invocation by SAML

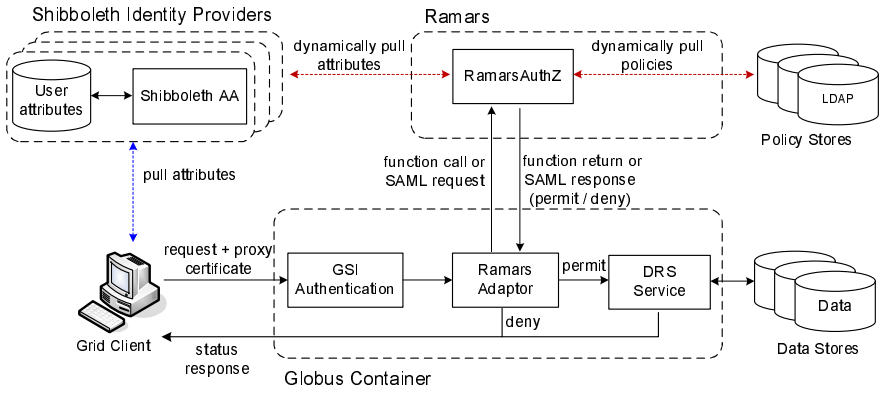


Figure 6.9: Integrated RamarsAuthZ Authorization System Overview

messages. In terms of attribute acquisition, both "push" and "pull" modes are supported. A Grid client can retrieve his/her attributes from the Shibboleth IdPs and "push" these attribute assertions upon the Globus DRS service invocation as legitimate extensions in the proxy certificate. In addition, the Grid clients can also embed the information about his/her preferred Shibboleth IdPs within the proxy certificate so that RamarsAuthZ is able to dynamically locate and retrieve the attributes for authorization. We call this proxy certificate with specialized extensions as a RamarsAuthZ proxy certificate.

The overall authorization flow for DRS service works as follows: the service provider of DRS first specifies and stores the ROA authorization policies in his administrative domain. The DRS service maintains the location reference of the ROA policies when the service is deployed in the Grid infrastructure. When a Grid client sends a RamarsAuthZ proxy certificate and his data replication request to the Grid DRS service, the client is authenticated through the Grid GSI module. Then RAMARS Adaptor is invoked to parse the extensions in the proxy certificate and prepare an authorization request for RamarsAuthZ service to check whether the Grid client is authorized to invoke the DRS service. The authorization request includes information on the requester's attributes and/or preferred IdPs passed by the proxy certificate, and the location reference of the service provider's ROA policies. Based on the authorization request, the RamarsAuthZ service can dynamically retrieve the service provider's ROA policies and the requester's attributes to make the authorization

decision. The decision is sent back to RAMARS Adaptor and enforced accordingly by the DRS service.

Substantial extensions and system modules have to be introduced to the original RAMARS system in order to enable the proposed functionalities and workflows. In the next several sections, we first share the experience in integrating RAMARS with Globus and Shibboleth. We then discuss the enhanced DRS service for data-level access control. The general trust management issues in our system are also discussed.

6.3.4 RAMARS-Globus-Shibboleth Integration Details

In our system design and implementation, we identify three key challenges and introduce multiple system components to solve these issues:

- How does a Grid client generate RamarsAuthZ proxy certificates to "push" attributes or convey preferred IdPs information to the DRS service for authorization?
- How does the RAMARS Adaptor formulate authorization requests and invoke RamarsAuthZ service with different deployment options?
- How does a RamarsAuthZ service operate to render an authorization decision?

To address the first question, a *Ramars-proxy-init* tool is introduced to facilitate a Grid user to create a RamarsAuthZ proxy certificate. In particular, a Grid user can collect SAML attribute assertions from his/her Shibboleth IdPs by using any client tools that can communicate with a Shibboleth IdP. Then the user triggers the *Ramars-proxy-init* tool to generate an X.509 proxy certificate including the attribute assertions as *SAMLX509Extension*. In case when the Grid user wants the system to operate in a "pull" mode, we also introduce another extension as *IdPInfoExtension* for the user to include a list of metadata information about his/her preferred IdPs. The IdP metadata contains important information for RamarsAuthZ to locate and communicate with an IdP, such as the IdP's unique provider ID, the endpoint location, the IdP's certificate for SSL communications, and so on. Such metadata can be easily retrieved from the Shibboleth IdP site or Shibboleth federations.

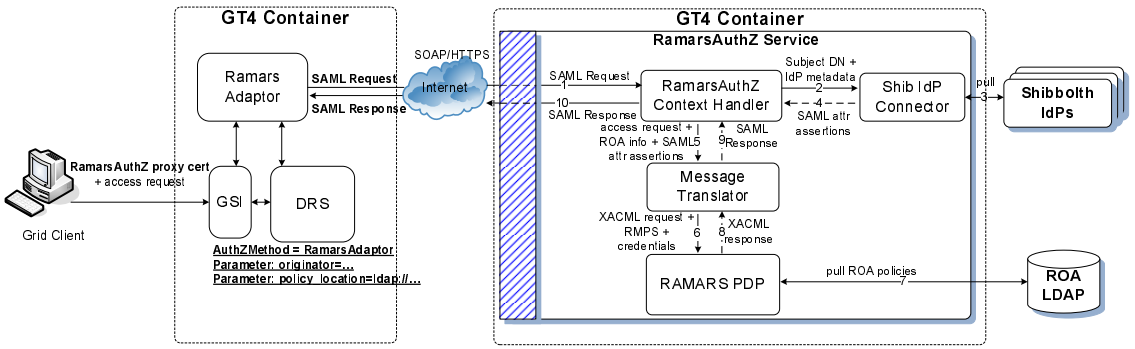


Figure 6.10: RamarsAuthZ Service

Since the proxy certificate is a self-issued certificate by the Grid user, all embedded SAML assertions and IdP's metadata must be signed by the issuing IdP and the metadata distributor, respectively. Any unsigned attribute assertions and IdP metadata without proper integrity protection are discarded and cannot be used by RamarsAuthZ for the authorization evaluation. By including these information in the legitimate extension fields of an X.509 certificate, the RamarsAuthZ proxy certificate can be accepted and verified by the GSI authentication module in GT4 without requiring any further changes.

To address the second question, the RamarsAuthZ service requires three sources of information to make authorization decisions: the access request to the Grid service, the location of the originator's ROA policies, and the requester's attribute credentials in the "push" mode or preferred IdPs information in the "pull" mode. The RAMARS Adaptor is the major component we plug into GT4 to convey these information. For now, we only focus on the service-level authorization and leave the data-level control to next section. The service provider of the Grid services, as the originator, is responsible to control the invocation of its services by notifying RamarsAuthZ its *ROA* policies. This is achieved by the Grid service provider to configure the originator information and the location of its *ROA* policies as RamarsAuthZ specific parameters at the time when the Grid service is deployed in GT4. And these parameters can be automatically passed on to the RAMARS Adaptor by GT4 when the Grid service is invoked. Upon receiving a Grid client's access request, the user's RAMARSAuthZ proxy certificate is received by the GT4 container as well. By

parsing the extension fields in the proxy certificate, the requester's credentials or IdPs information can be achieved. Thus, the RAMARS Adaptor can formulate the authorization decision query with all information required by the RamarsAuthZ service.

The invocation of RamarsAuthZ is fairly simple via function calls when RamarsAuthZ service is deployed locally. Yet it is more challenging when RamarsAuthZ service is deployed remotely as a Grid authorization service. In particular, the RamarsAuthZ service must be exposed through a standard SAML callout interface, and the SAML message is the only media for requesting and expressing authorization assertions and decisions from RamarsAuthZ service. According to GGF's SAML AuthZ specification [112], there are two SAML extensions being defined for message exchange between the calling service and the Grid authorization service:

- *ExtendedAuthorizationDecisionQuery*: Besides specifying the common elements in a SAML authorization decision query, such as the requested *Subject*, *Resource*, *Action* and *Evidence*, the *ExtendedAuthorizationDecisionQuery* allows the calling service to include an *AuthorizationAdvice* element to convey additional information such as where the remote Grid authorization service may obtain the subjects's attributes in the "pull" mode of operations. In addition, the *ExtendedAuthorizationDecisionQuery* also allows the calling service to specify whether a simple or full authorization decision needs to be returned by the Grid authorization service as a boolean value of the *RequestSimpleDecision* attribute.

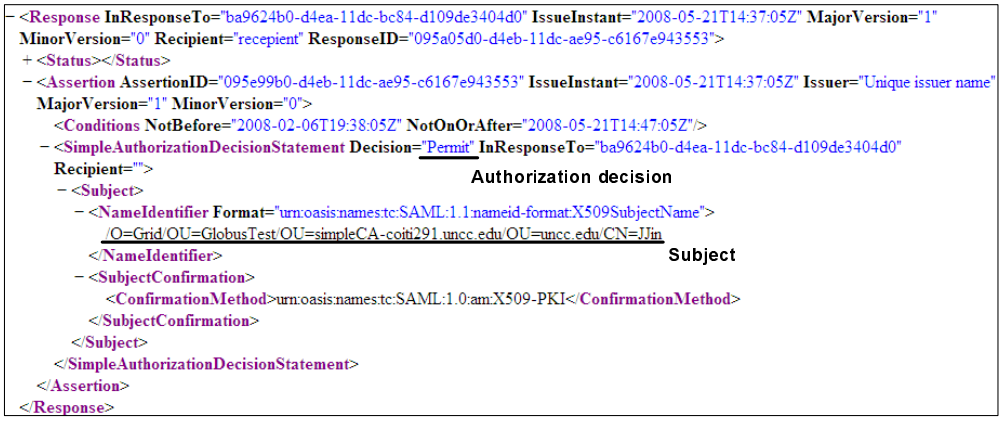
- *SimpleAuthorizationDecisionStatement*: When the calling service only requests a for simple decision (the *RequestSimpleDecision* attribute is set to *true* in the SAML request), a *SimpleAuthorizationDecisionStatement* is returned by the Grid authorization service without tedious enumeration of whole authorized actions. Such simpler response, in turn, could improve the efficiency for the calling service to process the response.

We further explore the *ExtendedAuthorizationDecisionQuery* (simply called *Query*) to include authorization related information for RamarsAuthZ service. In particular, a user's pushed SAML attribute assertions are directly included as *Evidence* in the *Query*. As the *ROA* policy information and the user's IdPs information are also important for RamarsAuthZ to render an authorization decision, they are encapsulated in the legitimate extension elements as *ROAReferenceAdvice* and *AAReferenceAdvice*, respectively. Figure 6.11(a) shows an example of SAML *Request* captured in our system for a user "*CN=JJin*" to request the DRS service, and Figure 6.11(b) shows a detailed attribute assertion issued by a Shibboleth IdP asserting that '*CN=JJin*" is affiliated with "*LIISP*" lab. The request is sent to RamarsAuthZ service to evaluate, and a "*Permit*" decision is sent back in a *SimpleAuthorizationDecisionStatement* as shown in Figure 6.11(c). With a "*Permit*" decision, the user "*CN=JJin*" is granted to invoke the DRS service.

To address the third question, we design RamarsAuthZ service as illustrated in the right portion of Figure 6.10. All communications to and from the service are through a standard SAML interface (step 1). The RamarsAuthZ Context Handler is responsible to parse the incoming SAML *Request*. Then Shib IdP Connector is invoked to initiate communications with the specified IdPs, one at each time, to acquire the user's attribute assertions (steps 2-4). The Message Translator is responsible to convert the messages from SAML to the XACML-based input that is accepted by the original RAMARS PDP. With conflicting message formats being transformed, the policy evaluation is carried out by the original RAMARS PDP without any additional changes (steps 5, 6, 8 and 9). And finally the authorization is returned back as a SAML *Response* to the Ramars Adaptor for decision enforcement (step 10). Yet from the originator's policy specification perspective, *ROA* policies have to be adjusted for the integrated Grid environments. Especially, since attributes are asserted from multiple Shibboleth IdPs, the trustworthiness of each attribute assertion should be based on the trust level an originator has assigned to the issuing Shibboleth IdP.

(a) SAML Request

(b) SAML Attribute Assertion



(c) Sample SAML Response

Figure 6.11: SAML Messages

6.3.5 Enhanced DRS for Access Control

Data resource originators delegate the data sharing responsibilities to the Globus DRS service, yet the DRS service should also be able to enforce access control on their behalf. The RAMARS framework is specialized to provide efficient access control for data sharing activities including the data discovery, data access, and data dissemination. We demonstrate such capabilities by implementing an enhanced DRS service where DRS acts as the PEP to enforce an originator's policies during each single step of the sharing operations. The data discovery functionality in DRS is performed by the RLS registry, where a *logical name–physical location* mapping is maintained for each data resource and its replicas. In order to enable the originator control, we add additional attributes associated with each RLS entry, namely *originator* and *roa_location*. With these two attributes being specified by the data originator, DRS service can not only discover the physical location of the data resource, but also collect the necessary information for the RamarsAuthZ service to locate the data originator's *ROA* policies.

In the enhanced DRS, the access control for *data discovery* is conducted when DRS receives the response from the RLS registry. An access request is generated to query RamarsAuthZ service whether or not the requester is authorized to "*query*" the physical location of the requested data file. A "Deny" decision results in the failure of file location, and all further data replication operations are aborted by the enhanced DRS. With a "Permit" decision, on the contrary, the enhanced DRS needs to further check with RamarsAuthZ to determine whether the requester is authorized to "*replicate*" the data, and a "Permit" decision would trigger the replication operation in RFT. After the data file is successfully replicated, a final *data dissemination* authorization request is sent to the RamarsAuthZ service to check whether the requester is authorized to further "*disseminate*" the replica to others. The replica is registered back in the RLS registry for other DRS queries only if the requester is authorized to do so. Figure 6.12 shows the complete workflow for the enhanced DRS to conduct the data replication and registration while enforcing originator's policies.

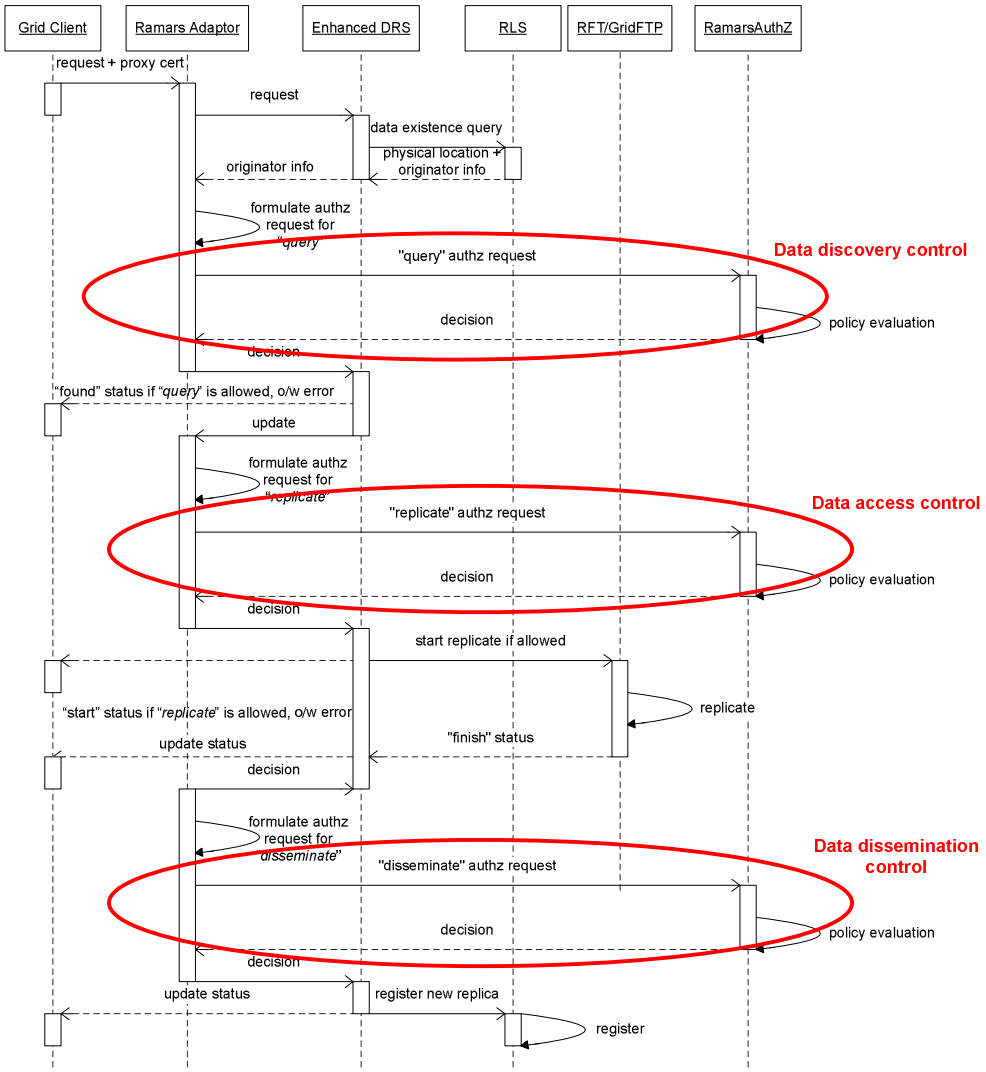
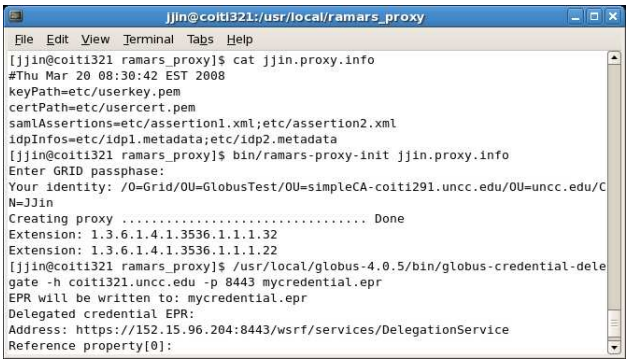
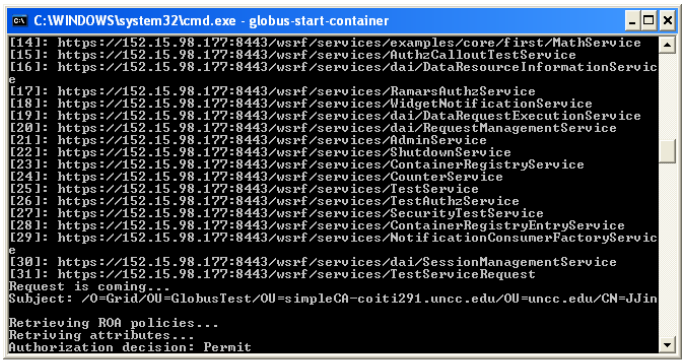


Figure 6.12: Sequence Diagram for Enhanced DRS Operations

Figure 6.13(a) shows a snapshot where a Grid client utilizes the *Ramars-proxy-init* tool to generate the RamarsAuthZ proxy certificate before invoking the enhanced DRS service. The proxy certificate includes two SAML attribute assertions and metadata of two preferred IdPs as extensions. Figure 6.13(b) shows a snapshot for the RamarsAuthZ service, deployed as the 17^{th} service in a GT4 container, to receive and process a SAML authorization request as shown in Figure 6.11(a).



(a) RamarsAuthZ Proxy Certificate Generation



(b) RamarsAuthZ Service and Policy Evaluation

Figure 6.13: Implementation of RamarsAuthZ Service

6.3.6 Trust management

Trust management in our proposed architecture is considered with two aspects: the organizational level concerning the necessary trust relationships between the involved parties and the technical level with respect to the implementation details.

At the organizational level, we consider four main entities, a Grid client, a Grid service provider (SP), the Grid resource providers (RPs) and the attribute providers (APs). In our system, the Globus DRS is a particular SP, the data resource originators are RPs and Shibboleth IdPs are APs. To simplify the analysis, we treat the RamarsAuthZ service as part of the SP as it provides authorization decisions for the SP to enforce. Since the Grid client does not directly interact with the RP, the RP has to rely on the SP to perform some of the authentication and authorization tasks. As a policy-driven approach, the RPs maintain their control through defining the *ROA* authorization policies, within which the

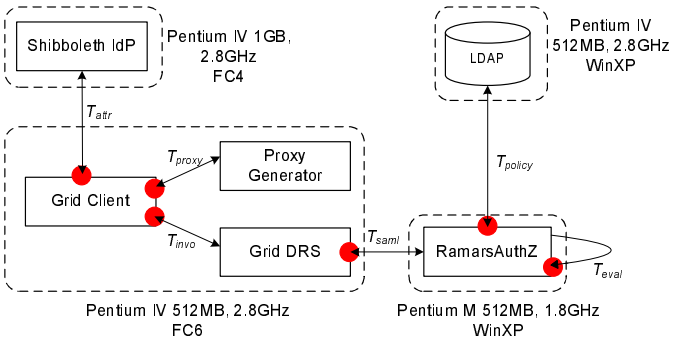
trust relationships between RPs and APs are explicitly defined as various trust levels in the Trust Assessment policies (TAP). However, the RPs still have to trust the SP to a considerable degree with which the authentication and *ROA* policies can be faithfully enforced. Additionally, the RPs also need to trust the APs for providing right attributes while these attributes are correctly handled by the SP associated with the RamarsAuthZ service.

At the technical level, the trust management among the Grid client, SP and RPs is carried out by the GSI infrastructure implemented in the Globus Toolkit where a Grid VO-CA is the centralized trust anchor. However, the APs leveraged by RamarsAuthZ for authorization is managed in Shibboleth environment. It is necessary to bridge the trust relationships between these two distinct environments. In our implementation, we adopt the GridShib plugin to query Shibboleth IdPs (Shib-IdPs) for user attributes. The trust relationship is based on a bilateral arrangement between the two parties by exchanging and consuming each other's metadata. In other words, the RamarsAuthZ service maintains a set of certificates identifying trusted IdPs for authentication and secure communication purpose, while those IdPs keep the certificate of RamarsAuthZ. Therefore, with n entities being involved, there are $O(n^2)$ bilateral relationships to be managed. With the Shibboleth federation being involved, a single trusted Shibboleth federation CA is introduced to all involved Shib-IdPs and Shib-SPs where the metadata are centrally maintained by the federation. This could partially ease the trust relationships managed by the Shib-IdPs and the RamarsAuthZ. In particular, it is possible for RamarsAuthZ to maintain two parallel certificates: one is issued by Grid-VO CA for authentication with all parties in Grid environments, and the other is issued by Shibboleth federation CA for the RamaraAuthZ to retrieve attributes from Shib-IdPs. In addition, other approaches such as bridge CA and online CA can be further explored for more seamless integration solutions on the trust relationships between Grid VO-CA and Shibboleth federation CA.

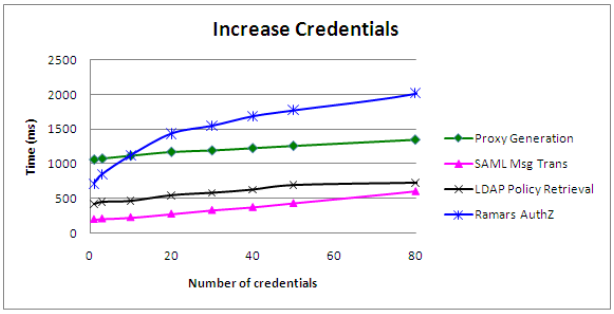
6.3.7 Performance Evaluation

We conduct a series of experiments to evaluate how well the system scales along with the increased evaluation complexity and also analyze the application overhead of RamarsAuthZ authorization service over Globus DRS service. In our testbed, the enhanced DRS service and RAMARS Adaptor are deployed within Globus Toolkit version 4.0.5 on a Pentium IV machine with Fedora Core 6. The *Ramars-proxy-init* tool is also deployed at the same machine for a Grid client to generate X.509 RamarsAuthZ proxy certificates and invoke the DRS service. The RamarsAuthZ service is employed within a WSRF-compliant WS Java container on a Pentium M machine with Windows XP. The Shibboleth IdP 1.3.3 is installed on a Pentium IV machine with Fedora Core 4. And IPlanet Directory Server is installed on a Pentium IV Windows XP machine as the back-end LDAP repository for the originator's *ROA* policies. All machines are located within the University's domain. We develop metrics to evaluate the performance of the system and the measurement of these metrics is performed by applying monitors at various locations of the system. Figure 6.14(a) illustrates our testbed and the monitors we put in our system based on the following metrics:

- *Attribute Retrieval Time (T_{attr})*: T_{attr} is the time taken by a Grid client to retrieve his/her attribute assertions from Shibboleth IdPs.
- *Proxy Generation Time (T_{proxy})*: T_{proxy} is the time taken by a Grid client to invoke $Ramars - proxy - init$ tool to embed attribute assertions and Shibboleth IdP information within a proxy certificate.
- *DRS Invocation Time (T_{invo})*: T_{invo} is the time taken by a Grid client to invoke the enhanced Globus DRS service for completing a file replication task.
- *SAML Message Transfer Time (T_{saml})*: T_{saml} is the time taken by the Globus DRS service to send out an authorization request to RamarsAuthZ service and get the response back over SAML message exchange protocol.



(a) RamarsAuthZ System Evaluation Testbed



(b) RamarsAuthZ System Performance Evaluation Result

Figure 6.14: RAMARS System Evaluation

- Policy Retrieval Time (T_{policy}): T_{policy} is the time taken by the RamarsAuthZ service to retrieve the originator's *ROA* policies from LDAP policy store.
- Policy Evaluation Time (T_{eval}): T_{eval} is the time taken by the RamarsAuthZ service to make the authorization decision based on the originator's *ROA* policies.

As a Grid client's authorization privileges are determined by his/her attributes, the client's attribute assertions need to be transferred all the way from the client through the Globus DRS service to the RamarsAuthZ service for making authorization decisions. In this sense, the scalability of the system is largely affected by the number of attribute assertions handled by the system. Therefore, our first experiment is conducted by increasing the number of attribute assertions [3]. Figure 6.14(b) indicates the testing result as the number of attribute assertions gradually increases. In particular, SAML message transfer time T_{saml}

[3] Shibboleth IdP can only assert very limited number of user attributes. In our experiment, we assume the Grid client already has the attribute assertions stored locally. And the performance evaluation of Shibboleth IdP (T_{attr}) is beyond the scope of this research.

and proxy generation T_{proxy} increase linearly as they have direct associations with the size of attribute assertions. The increase of attributes, however, has an indirect effect on the size of the originator's *ROA* policies. Therefore, LDAP policy retrieval T_{policy} increases with a flatter rate. The RamarsAuthZ policy evaluation T_{eval}, on the other hand, shows a polynomial trend with the increase of policy evaluation complexities, which is a desirable property that makes the system more scalable to a large number of attributes.

Our next experiment is conducted to analyze the overhead of RamarsAuthZ with the Globus DRS data sharing service. In particular, not only the DRS service itself needs to be authorized, but also the data replication operations require a series of fine-grained authorizations for each step of data replication. The overhead introduced to achieve such fine-grained authorizations should be measured. We use the same sample data file of size 121,781KB as we conducted for ShareEnabler evaluation to be replicated in Globus DRS service for our experiment. The base time for DRS invocation time T_{invo} is measured by applying the default GridMap authorization. We deploy RamarsAuthZ as a remote SAML-enabled authorization service to measure the overhead of the one-step authorization for DRS service, and then measure the fine-grained multiple-step authorizations for DRS operations. In addition, to better understand the overhead of standard SAML authorization protocols, we deploy RamarsAuthZ as a local authorization module so that SAML message exchange is replaced by a local procedure call. Table 6.2 shows the detailed results of our experiment when we adjust the number of attributes from 1 to 80. The results are measured in milliseconds and computed based on the average of 100 test runs. With the extreme complexity of evaluation, the one-step RamarsAuthZ authorization for DRS service introduces less than 8% overhead compared to the traditional GridMap authorization, which we believe is a promising outcome with respect to the performance of RamarsAuthZ authorization service. Same to our expectation, achieving a fine-grained authorization for stepwise data sharing involves considerable cost. However, considering the potential reduction of the administrative overhead against the practices of manually maintaining individual

Table 6.2: RamarsAuthZ Overhead Analysis

Attr #	**Base**	**For DRS Service**		**For DRS Operations**		**Local Module**	
	Time (*ms*)	Time (*ms*)	Overhead	Time (*ms*)	Overhead	Time (*ms*)	Overhead
1	42584	43910	3.11%	45614	7.12%	44834	5.28%
3	42584	44081	3.52%	45948	7.90%	45132	5.98%
10	42584	44388	4.24%	46598	9.43%	45706	7.33%
20	42584	44839	5.30%	47530	11.61%	46434	9.04%
30	42584	45042	5.77%	48021	12.77%	46709	9.69%
40	42584	45272	6.31%	48527	13.96%	47043	10.47%
50	42584	45481	6.80%	49006	15.08%	47294	11.06%
80	42584	45931	7.86%	50256	18.02%	47856	12.38%

user accounts, RamarsAuthZ service still shows clear advantages both architecturally and technologically. Compared to the locally deployed authorization module, the overhead of SAML authorization messages cannot be neglected. Therefore, the usage of SAML for authorization in Grid environments needs to be limited for simple and optimized message assertion exchanges. Especially, instead of transferring a large number of attribute assertions as "push" mode, a reference to the Grid client's IdPs should be transferred within SAML message for RamarsAuthZ to operate under "pull" mode.

In order to further justify our approach, we compare our RamaraAuthZ system with a number of existing authorization systems proposed for Grids environments, including Globus Community Authorization Service (CAS) [92], Virtual Organization Membership Service (VOMS) [11], Akenti [108] and PERMIS [27]. According to the access control challenges as we identified in Section 6.3.1, we compare the systems from the following aspects:

- Originator control: to examine whether the authorization system provides facility for the resource owner to control the access.
- Attribute-based access control: to examine whether the authorization system establishes authorization based on user attributes.
- Policy engine: to examine whether the authorization system consists of a policy engine for complex policy evaluations.

Table 6.3: Comparison With Existing Grid Authorization Systems

	CAS	**VOMS**	**Akenti**	**Permis**	**RamarsAuthZ**
Originator control	No	Partial	Yes	Partial	Yes
Attribute-based access control	No	Yes	Yes	Yes	Yes
Policy engine	No	No	Yes	Yes	Yes
Policy store	Centralized, single	Centralized, single	Distributed, multiple	Centralized, single	Distributed, multiple
Supported IdPs	N/A	Multiple	Multiple	Single	Multiple
Policy and IdP configuration	Static	Static	Static	Static	Dynamic
Operation modes	Push	Push	Pull	Pull	Push, pull
Service-level control	Yes	Yes	N/A	Yes	Yes
Data-level control	Yes	Maybe	N/A	No	Yes

- Policy store: to examine the operation mode of the policy store for the authorization system.
- Supported IdPs: to examine the number of IdPs that could be configured with the authorization system.
- Policy and IdP configuration: to examine whether the authorization system relies on static configurations for policy and attribute management.
- Operation modes: to examine the policy and attribute retrieval mode of the authorization system.
- Service-level control: to examine whether the authorization system supports service-level control to protect Grid services.
- Data-level control: to examine whether the authorization system supports data-level control to protect data resources being process by Grid data management services.

Table 6.3 summarizes comparisons between our proposed RamarsAuthZ system with the above-mentioned authorization systems. It is evidently shown that RamarsAuthZ system has advantages in various aspects such as established access control model and policy, system architecture and deployment, and the protection ranges.

Summary: In this Chapter, we introduced the design and implementation of the RAMARS authorization system. We demonstrated the feasibility of our proposed system by implementing and evaluating proof-of-concept prototypes – ShareEnabler and RamarsAuthZ systems – to support secure information sharing in P2P and Grids environments, respectively. In Chapter 7, we elaborate a case study as a feasible application of RAMARS in healthcare domain to support the sharing of medical information. Especially, we introduce a substantial extension to allow patients to define fine-grained policies to authorize and selectively share sensitive medical information.

CHAPTER 7: CASE STUDY IN HEALTHCARE

Healthcare environments involve a wide range of individuals and organizations with diverse perspectives within the healthcare process. The recent endeavor of the Nationwide Health Information Network (NHIN) initiative [60] requires a nationwide electronic communication infrastructure being established to allow patients, physicians, hospitals, public health agencies, and other authorized users to share clinical information in real-time and authorized manner. This coincidently reflects the motivation scenario of our proposed RAMARS framework. Therefore, healthcare is one of the perfect application domains where RAMARS may play a major role to enable secure and authorized medical information sharing. In particular, our RAMARS framework leverages role-based approach to achieving effective and manageable authorization, and promotes attribute-based role assignment and trust evaluation to dynamically identify unknown users for data sharing. These are the essential features, as we have demonstrated through this entire dissertation work, to be incorporated in secure healthcare applications for the originators of the medical information to effectively manage access control among distributed healthcare stakeholders for sharing sensitive medical information.

Nevertheless, sharing of medical information is an important and challenging issue. Patient privacy concerns, along with threats that could expose sensitive medical information, highlight the need for security and privacy technologies to be well-integrated into the healthcare system so that we can ensure access to sensitive information is limited only to those entities who meet a legitimate need-to-know requirement. For instance, a patient's medical information pertaining to an HIV/AIDS diagnosis may be explicitly hidden from surveillance data sharing unless a specific monitoring option is indicated. Considering this special requirement, the medical information itself must be further analyzed and classified

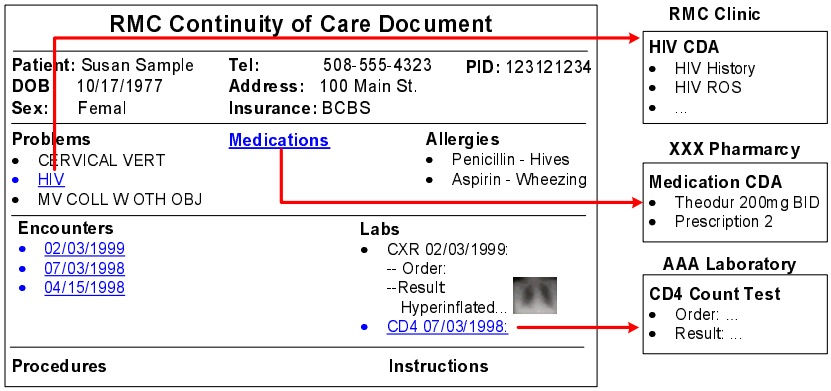


Figure 7.1: Motivation EHR Document

in order for the originator to define fine-grained RAMARS policies being applied to the entire medical data set or a partial data set. In this Chapter, we conduct a case study to explore the necessary elements and policies for RAMARS to support the sharing of medical information.

7.1 Access Control Challenges in Sharing EHRs

The adoption of electronically formatted medical records, so called Electronic Health Records (EHRs), has become the primary concern for a broad range of health information technology applications and practitioners. In order to better illustrate the special access control challenges on sharing of EHRs, we consider a typical clinical EHR document and we demonstrate our proposed approach using the same EHR document throughout our case study.

Suppose RMC Clinic is a member of a regional healthcare collaboration network, where health information can be exchanged through an established infrastructure with other collaborating institutions. Figure 7.1 illustrates a sample Continuity of Care Document in the clinic for a patient named Susan Sample [36]. The EHR document includes Susan's demographics, medical problems, medications, labs, and so on. The medical information is recorded in various data types such as texts, numbers and images. Some fields inside the document may refer to other external clinical documents. For example, Susan's HIV/AIDS disease history may be maintained in another folder of the patient, and Susan's current

medications may be directly linked to the records operated by her pharmacist.

The example clinical document has demonstrated the unique characteristics of an EHR including the composition of various data types and connections among different pieces of information from multiple sources. Given the complexity of EHR documents, the contained information should be legitimately exchanged to satisfy needs of different parties within the collaboration network. In particular, the lab orders need to be communicated with appropriate laboratories and specific test codes are used to trigger the billing process. The doctor's prescriptions, on the other hand, are necessary to be filled by the pharmacist, and proper referrals are exchanged with specialists for complex medical problems. In other words, the need-to-know principle must be strictly enforced for each responsible party to obtain only the necessary information to carry out its task. This requires the data originator to distinguish different portions of the document and define fine-grained RAMARS policies to control the selective sharing of EHRs. In order to achieve this goal, it is essential to explicitly identify the data structure within an EHR document, and the data items inside the document must be classified with regard to different sources, data types, and sensitivity levels to guide the selection of specific parts with various protection granularity levels within the document. We thus propose an approach with a series of complementary policies for an originator to modeling the selection and authorization of EHRs. Our model first introduces a level of abstraction to formulate the logical structure of an EHR document in terms of its internal data items under protection and relationships among them. The data items within the structure are categorized by three dimensional properties – origin, sensitivity classification and object type – to facilitate the authorization model and accommodate the protection requirements. By manipulating the selection criteria of these properties, different subsets of data items within the document can be dynamically selected to apply RAMARS policies.

The rest of this Chapter is organized as follows. In Section 7.2, we provide a brief background overview of the emerging EHR standards and existing security solutions for EHR systems. In Section 7.3, we introduce the the logical model to formulate the semantics

and structure of EHRs, and discuss the access control policy scheme around the logical structure. Finally in Section 7.4, we introduce the design and implementation of our proof-of-concept EHR sharing system – *InfoShare*.

7.2 Related Work

EHR Standards: There are several standards currently under development to specify EHRs, such as openEHR [87] and Health Level 7 (HL7) Clinical Document Architecture (CDA) [57, 36]. These standards aim to structure and markup the clinical content of an EHR for the purpose of exchange. The most important concept introduced in openEHR is the *archetype*, which is used to model healthcare concepts such as blood pressure and lab results. These archetypes serve as fundamental building blocks to form various clinical EHR documents. Meanwhile, these archetypes and the contents contained in them should be legitimately distinguished and authorized in the process of information exchange across healthcare systems. Similarly, CDA defines the structure and semantics of medical documents in terms of a set of coded components (called vocabulary) to model basic medical concepts.

A common feature of all emerging EHR standards is that the clinical concepts are modeled and expressed independently from how the data is actually stored in the underlying database. By implementing or converting to the EHR standards, a “common language" is established between different medical information systems to communicate and share standardized medical information with each other. Therefore, instead of being carried out at the lower database level, data selection and authorization should be defined and enforced with common understanding of EHR standards.

Access Control for EHR Systems and e-Consent Systems: A number of solutions have been proposed to address the security and access control concerns associated with EHR systems. In [26], the authors propose a set of authorization policies enforcing role-based access control for the electronic transfer of prescriptions. In [39], the paper demonstrates

an implementation of EHR prototype system including a basic network and role-based security infrastructure for the United Kingdom National Health Service. In [21], the paper presents a policy-based security management framework to enforce context based authorizations for federated healthcare databases. Role-based access control has become the common theme applied in these approaches. However, as illustrated in our RAMARS framework, RBAC alone cannot effectively authorize the unknown users across domains. This has posed serious scalability limitations for these systems to be applied in supporting the sharing of EHRs in large-scale distributed healthcare collaborations. In addition, the EHR considered in these approaches is either a general abstract object or an isolated primitive object. None of these approaches took into account of the complex structure of EHR documents, and thus cannot support a more fine-grained access control as illustrated in the above-mentioned example.

Achieving privacy preservation in medical information sharing is a critical concern for an EHR system. Several purpose-based access control models have been proposed recently to control access to the sensitive data [24, 114]. These models associate the intended purpose with a given data element, and the access purpose should be consistent with the data element's intended purpose. As healthcare is such a complex domain where a patient's EHRs need to be shared among several parties with different duties and objectives, the purpose-based access control alone cannot meet all the patient's privacy protection requirements. In our case study, we consider additional deciding factors beyond purpose to control the selective sharing of EHRs in a more flexible and effective way.

As the patient is considered as the ultimate owner of his medical information, the essence of originator control highlights the need for patients to control the sharing of his own medical information between healthcare providers. "e-Consent" mechanisms allow patients to issue or withhold authorization policies as electronic consents to those who wish to access their electronic health information [30, 100, 86, 94]. Several consent models with associated consent templates have been identified [30, 100], and a few e-Consent based

systems have been built upon these guidelines in Europe and Australia [94, 86]. However, it is still essential to develop a systematic approach to determining how a patient's consent is expressed and at what granularity the consent is applied to which portion of the EHRs.

Authorization Models for Structured Data: Authorization of EHRs requires clear understanding of the internal data items/clinical concepts and their structural relationships. There has been a considerable amount of work in regulating access to structured or semi-structured data.

The access control models proposed in [43] and [95] are especially tailored to object-oriented databases storing conventional structured data, where information is represented in the form of objects. These models consider a rich semantic structure of objects incorporating inheritance, aggregation, and composition associations. The relationship of objects in the database is modeled as a hierarchical structure so that the validity of an authorization rule written at some level can be efficiently propagated to its descendants. Such features can be adopted in modeling the logical structure of EHRs. However, these models have several shortcomings. On the one hand, EHR documents are stored and exchanged based on standards, which are defined independently from underlying database structures. The object relationships and navigational patterns defined in standards may be totally different from the ones enforced by access control mechanisms. On the other hand, as identified in our motivation scenario, the medical information may be distributed at different sites. This unique feature cannot be addressed by a localized object-oriented database.

XML has become the de facto mechanism for sharing data between disparate information systems. It is essentially adopted by HL7 to carry out its standardization efforts to describe, store and exchange health records. Regulating access to XML documents has attracted considerable attentions in recent years [20, 31, 50]. All these work represent an XML document as a hierarchical tree structure and its authorizations are propagated along with the association links to achieve different granularity levels. However, all these approaches define access control rules for particular elements and attributes of an XML doc-

ument. The selection of a collection of data elements requires a number of authorization rules to be defined and evaluated. This is obviously not effective and efficient in practice to authorize and share a specific part of the document for fulfilling the specific functional purpose of the requesting party. In addition, an XML document itself is not semantically enough to represent a variety of data types as encountered in EHRs (i.e., image, audio and video). Thus the access control mechanisms proposed for XML documents cannot meet the special requirements for sharing EHRs.

7.3 Authorization for Sharing EHRs

7.3.1 Logical EHR Model

In our model, an EHR document is logically modeled as a labelled hierarchical structure. The nodes represent the clinical data elements that need to be protected for sharing. Their relations are captured as the association links between the nodes within the hierarchy. Each node is associated with specific properties to address essential features regarding the sources of data and their sensitivity levels. The properties can be categorized into three dimensions: *origin*, *sensitivity*, and *object type*. The *origin* property is specified to indicate the source(s) of data within the EHR document. For instance, the information of *CD4* lab test in Figure 7.2 comes from *AAA* laboratory. The *sensitivity* property is designed to label a node based on the sensitivity of the content contained in it, which eventually can be used to prevent the patient's sensitive medical information from being disclosed unintentionally. In the practice of Iowa HISPC [63], the sensitivity classifications of medical data include general medical data, drug and alcohol treatment, substance abuse treatment, mental health, communicable disease (HIV, STDs, etc.), decedent, immunizations, and so on. Based on these classifications, the data elements representing the patient's *HIV* history and *CD4* lab test should be marked with a property of "communicable disease" ("*HIV*" for simplicity). The *object type* property gives another dimension on data node selection and protection. The nodes can be primitive types such as plain texts, dates and images. They can also be a

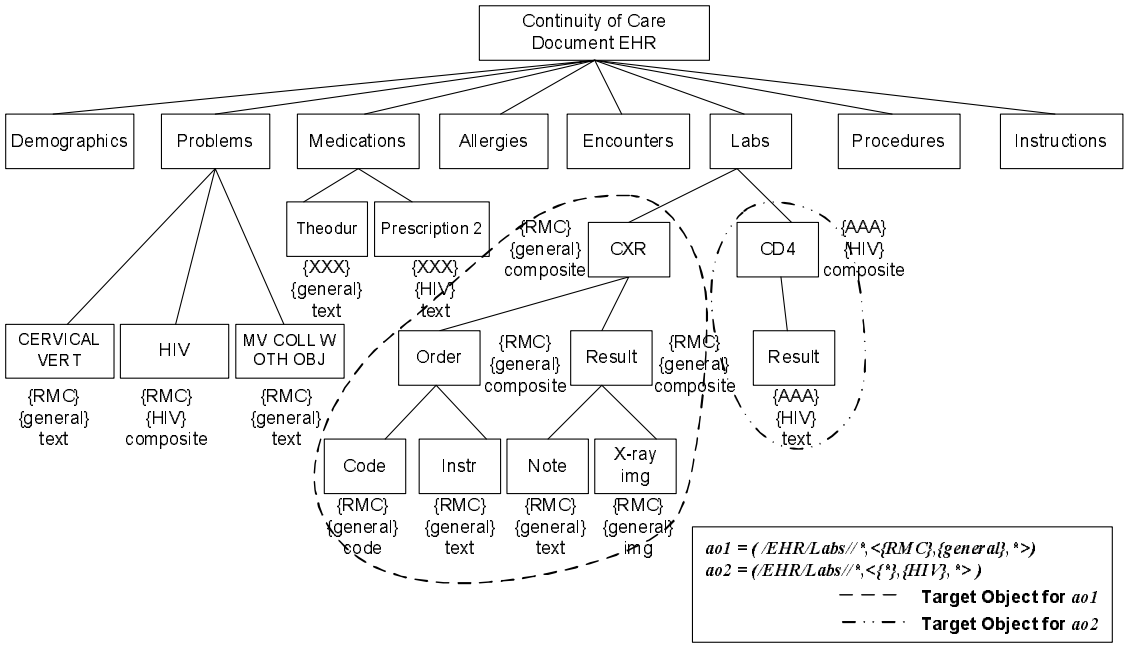


Figure 7.2: Virtual Composite EHR in a Hierarchical Structure

composite type in the hierarchical structure including other types of data nodes. Formally, an EHR can be uniformly modeled and defined as follows:

Definition 6 *(***Logical EHR Model***).* An EHR is a tuple $C = (v_c, V_o, E_o, \tau_{V_o})$, where

- v_c is the root representing the whole EHR object;
- V_o is a set of nodes within the hierarchical structure;
- $E_o \subseteq V_o \times V_o$ is a set of links between nodes; and
- $\tau_{V_o} : V_o \rightarrow P$ is a node labelling function to specify the property of a node. P is a set of properties defined in Definition 7.

Definition 7 *(***Property***).* Let O, S, and T be the sets of data origins, sensitivity classifications, and object types, respectively. And let $n = |V_o|$ be the number of nodes in an EHR structure C.

- $P_o = \{po_1, \ldots, po_n\}$ is a collection of origin sets, where $po_i \subseteq O$ is a set of origins associated with a node, $i \in [1, n]$;

Table 7.1: Path Expression for Node Selection

Expression	Description	Example
nodename	Select the named nodes	*CXR*
/	Select the node through absolute path from root node	/EHR/Labs/CXR
//	Select the node through relative path	//Labs/CXR
*	Select all immediate children nodes	//Labs/CXR/*
//*	Select all descendant nodes	//Labs/CXR//*

- $P_s = \{ps_1, \ldots, ps_n\}$ is a collection of sensitivity classification sets, where $ps_i \subseteq S$ is a set of sensitivity classifications associated with a node, $i \in [1,n]$; and
- $P = P_o \times P_s \times T$ is a set of three dimensional properties of origin, sensitivity, and data type.

Given a node $v_i \in V_o$ inside an EHR structure C, the function $\tau(v_i) = p$ retrieves the property label p for the node. And we use the dot notation to refer to a specific property dimension. For instance, $p.po$ refers to the data origin property; $p.ps$ refers to the sensitivity property; and $p.t$ refers to the object type. Within a logical EHR structure, nodes can be explicitly denoted by their identifiers, or can be implicitly addressed by means of *Path Expressions*. We apply an XPath-like expression for the path representation. Table 7.1 describes the notions and examples we use to select nodes inside an EHR illustrated in Figure 7.2.

7.3.2 Complementary Policy Specification

Our policy specification scheme is built upon the identified logical EHR model so that RAMARS policies can be effectively defined at different granularity levels within the structure. In addition, to accommodate the special privacy protection requirement, the purposes of sharing EHRs must also be explicitly specified. Therefore, the fundamental question towards the selective authorization of an EHR is to which portion of an EHR document RAMARS policies can be applied for what purposes of sharing. The role of the complementary policies is then to articulate and determine the selected objects with intended purposes of use within an EHR structure, and these selected objects would eventually con-

tribute to the *Resource* element specification in the Root Meta Policy Set (RMPS) for the entire RAMARS policies to apply.

In our hierarchical EHR model, XPath-like path expressions are utilized to specify the *scope* of data elements to which an authorization policy applies. Meanwhile, the *filtration properties* are defined to be compared with the property label of each node within the EHR, and only matched nodes are selected as the *Target Objects* of the RAMARS authorization. We formally define these concepts as follows:

Definition 8 *(***Filtration Property***).* Let O, S, and T be the sets of data origins, sensitivity classifications, and object types, respectively as defined in Definition 7. A filtration property is specified as a tuple $prop =< po, ps, pt >$, where $po \subseteq O$ is the filtration property for origins; $ps \subseteq S$ is the filtration property for sensitivity classifications; and $pt \subseteq T$ is the filtration property for object types.

Definition 9 *(***Property Match***).* Suppose $prop =< po, ps, pt >$ is a filtration property specification, and $p' = (po', ps', t')$ is the property label of a node, the node matches the filtration property if the following conditions are satisfied:

1. $p'.po' \subseteq prop.po$;
2. $p'.ps' \subseteq prop.ps$; and
3. $p'.t' \in prop.pt$.

Definition 10 *(***Object Specification***).* Let *scp_expr* be a scope expression to denote a set of nodes within the composition, and *prop* be a filtration property specification, the object selection specification is defined as a tuple $ao = (scp\_expr, prop)$. Given an EHR logical model $C = (v_c, V_o, E_o, \tau_{V_o})$ and an object selection specification ao, we define a function: $select(C, ao) \rightarrow V_a$, where $V_a \subseteq V_o$, to select the matched nodes within the specified scope as the Target Objects.

In specifying filtration properties, we also allow patterns to be used. Pattern “*" is to

indicate *any value* within a property dimension, and pattern "{*}" is to specify *any set* within a property dimension.

Example 1 The followings are two examples of object selection specifications against the EHR structure in Figure 7.2.

ao1: $ao1 = (/EHR/Labs//*, <\{RMC\}, \{general\}, *>)$; and

ao2: $ao2 = (/EHR/Labs//*, <\{*\}, \{HIV\}, *>)$.

The two object specifications select the same scope as the *Labs* in the EHR structure. ***ao1*** selects the nodes that come from *RMC* with *general* level of sensitivity and *any* object types. ***ao2*** selects the nodes from *any* origins with *HIV* level of sensitivity and *any* object types. Figure 7.2 illustrates the target objects being selected according to the $select()$ function in Definition 10 as two dashed zones. In particular, ***ao1*** results in the *CXR* lab test and all its children nodes being selected, and ***ao2*** results in the CD4 lab test and all its children nodes being selected.

To further address the privacy concerns in sharing medical information, an attribute of "purpose" is necessary to be specified in the authorization policy to confine the intended purposes/reasons for data access in healthcare practice. According to [35], business practices for health information exchange can be organized by 11 purposes including payment, treatment, research, and so on. Formally, intended purpose is specified as follows:

Definition 11 *(***Intended Purpose***)*. Let P be a set of purposes for business practices in healthcare domain. And let m be the total number of authorizations in the system. The intended purpose set $P_p = \{pp_1, \ldots, pp_m\}$ is a collection of possible intended purpose sets, where $pp_i \subseteq P$ specifies the intended purposes for a particular authorization, $i \in [1, m]$.

To summarize the above-mentioned policy elements, we could introduce the overall definition of an access control policy. For simplicity, we consider the users are authorized through their roles, and we use "*ramars-priv*" to indicate the detailed authorizations to the role that are determined by RAMARS ROA policies.

Definition 12 *(***Access Control Policy***).* Let R be the system-wide set of roles in a healthcare system, and ramars_priv be the RAMARS authorizations being assigned to the role. An access control policy is a tuple $acp =< role, ao, pp, ramars\_priv >$, where

- $role \in R$ is a specific role in the system;
- ao is an object selection specification resulting in a set of nodes $V_a \subseteq V_o$ being selected as target objects;
- $pp \in P_p$ is the intended purposes; and
- *ramars_priv* is the originator's authorizations being assigned to the role.

Example 2 Let $ao1$ and $ao2$ be specified as same as those in Example 1, the following access control policies can be articulated :

P1*:* $(\text{“}GP'', ao1, \{treatment\}, \{query, access\})$; and

P2*:* $(\text{“}SP'', ao2, \{treatment, research\}, \{access\})$.

In **P1**, the general practitioners (*GP*) are authorized to query and access the patient's *general* lab tests from *RMC* for the *treatment* purpose. In **P2**, the specialists (*SP*) are authorized to access the patient's *HIV* related lab tests for *treatment* and *research* purposes.

In healthcare practice, a default policy may be established to satisfy most patients' privacy requirements. Once a patient understands the default policy and agrees that it meets his/her needs, the patient may not need to further specify any specific access control policies to control the sharing of his/her medical information. In particular, HIPAA regulations are widely adopted by healthcare practitioners in the United States. With the agreement of the default setting, HIPAA generally allows health care providers to share clinical information without the individual's explicit permission for treatment, payment and health care operations [94]. In addition, in order to accommodate the emergency situations, a "break-of-glass" policy ("*BG*" policy for simplicity) should be specified to allow staffs in emergency rooms to access the patient's medical information without the patient's explicit

authorizations. Both the default policy and "*BG*" policy can be specified conforming to our policy schema.

Example 3 The default policy and BG policy can be specified as follows:
$\boldsymbol{P_D}$: $(\text{“}HP'', (\{*\}, \{*\}, *), \{treatment, payment, HCO\}, \{query, access\})$; and
$\boldsymbol{P_{BG}}$: $(\text{“}ERStaff'', (\{*\}, \{*\}, *), \{treatment\}, \{query, access\})$.

7.4 EHR Sharing System – *InfoShare*

We have designed and implemented a proof-of-concept system, called *InfoShare*, which is a simplified clinical information sharing system that utilizes our proposed model for a patient to control access of his/her medical information. In particular, *InfoShare* collects the patient's access control policies as patient consents, and uses these consents to selectively share the patient's medical information with different requesting parties through a point of service (POS) web application.

Architecturally, health information as EHRs is maintained and managed at each geographically distributed care provider's site with the notation of federation. The data elements in each EHR document should be associated with special properties for data filtration and authorization purposes. We therefore establish an EHR data labelling system in the federation to facilitate the property labelling for the care providers' EHRs. Our *InfoShare* system serves as a middleware application to create the logical structure of an integrated EHR document connecting a group of care providers and organizations within the federation. Meanwhile, *InfoShare* also serves as a gatekeeper to enforce the patient's authorizations to selectively share the patient's medical information. Figure 7.3 illustrates the overall architecture of an integrated *InfoShare* information system. As a general clinical information sharing system, the Registry Service, EHR Data Service, General Security Service and Health Information Communication Bus are common system components to achieve the required functionalities of secure data retrieval, logical EHR structure creation and communication with requesting POS applications. Especially, we inject the Consent Management Service, Policy Service, and EHR Authorization & Selection Service as the

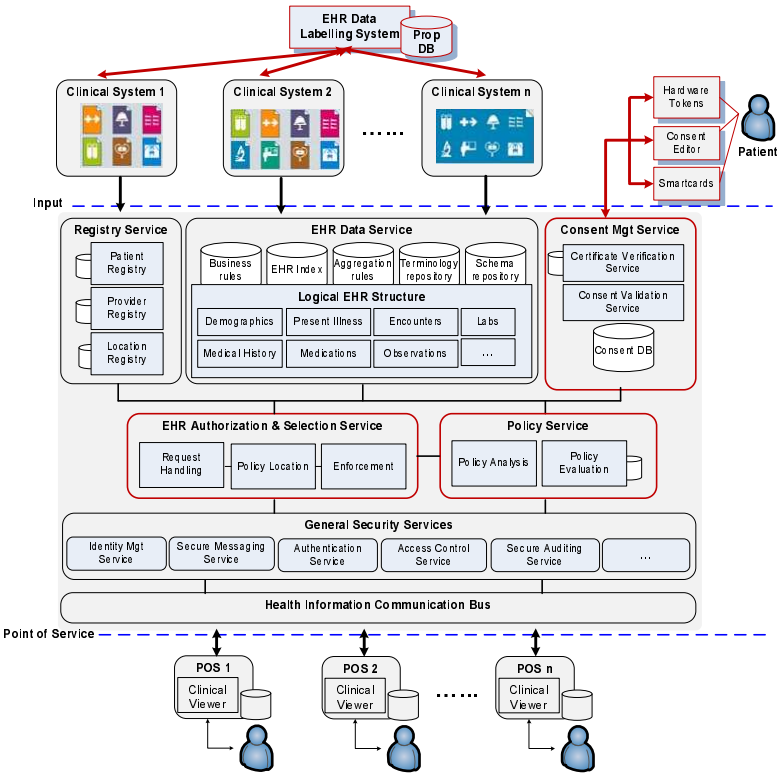


Figure 7.3: InfoShare System Architecture

major system modules to convey the core features of our proposed approach. In particular, the Consent Management Service enables the patient control by collecting and verifying the patient's access control policies as encapsulated in consents. A web-based consent editor tool is implemented to facilitate a patient to edit his/her policy consents, and interact with the Consent Management Service for the patient to store and update his/her consents. Besides, the patient consents can also be directly inserted into the Consent Management Service by the patient using smart cards or other hardware tokens. The Policy Service serves as the PDP to evaluate the patient's access control policies and derive the authorized medical data. The EHR Authorization & Selection Service is responsible to handle the data access requests, and enforce the access control decision by only sharing the authorized data with the requester.

In our *InfoShare* system, the expressed access control policies are encapsulated as consents. There are three types of consents in the system: the patient's specific consents, the

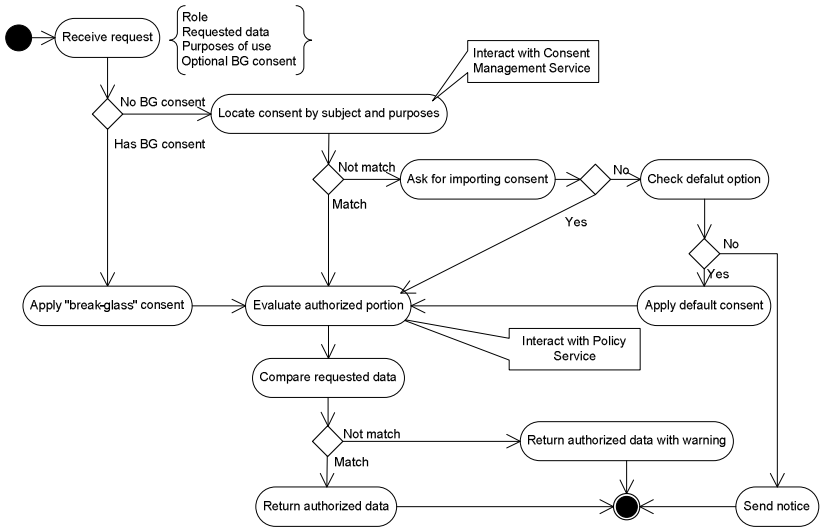


Figure 7.4: Consent Evaluation Procedure

default consent, and the "*BG*" consent, where the default consent and "*BG*" consent specify the default policy P_D and "*BG*" policy P_{BG}, respectively as shown in Example 3. The precedence order of evaluating these consents is defined as $BG\_consent \succeq patient\_consent \succeq default\_consent$. Figure 7.4 illustrates the procedures for the EHR Authorization & Selection Service to handle access requests and derive the authorized data to be shared. An access request includes information of the requester's role, the requested data, the intended purposes of use, and an optional "*BG*" consent in emergency situations. The "*BG*" consent in emergency has the highest priority in execution, therefore such consents are directly evaluated to get the authorized data. In other situations, the authorization service interacts with the Consent Management Service to locate the related patient consents based on the specified subject and intended purposes. If certain matched consents are located, the Policy Service is invoked to evaluate and derive the authorized portion of data. If there are no patient consents being located, our system asks for the patient to insert a patient consent at run-time and the consent is evaluated accordingly. Otherwise, the default consent is evaluated to derive the authorized data. After the authorized data portion is determined, it is compared with the requester's requested data and only matched data portion is returned to the requester. If the requester is not authorized to access all the data he requested, the

requester is notified with a warning, so that the requester may further ask for new patient consents to access the data for the need of medical practice. Such an effective mechanism is utilized to balance the data integrity concern of the practitioners and the privacy concern of the patients for shared EHRs.

In terms of implementation, the standard HL7 Clinical Document Architecture (CDA) [36] is utilized in our *InfoShare* system for the formal representation of EHRs. We use Jaxe XML editor [2] as the EHR data labelling service to associate properties with data elements in CDA EHRs. We implement the patient consents as X.509 attribute certificates [58], where the access control policies are encapsulated as attributes within the certificate. The *InfoShare* system implements a Java Servlet based web portal as the POS application for a healthcare practitioner to query and view the authorized medical information of a patient. Figure 7.5(a) shows an interface of the *InfoShare* POS portal for a practitioner to view a patient's integrated medical information, where only authorized information is displayed. Figure 7.5(b) shows the interface of the Consent Editor for a patient to edit his/her policy consents.

Summary: In this Chapter, we elaborated a case study to demonstrate a possible application of RAMARS to support fine-grained authorization and selective sharing of EHRs in healthcare domain. We introduced a logical EHR model to facilitate the selection of different portions of data within an EHR document for the originators or healthcare stakeholders to apply RAMARS policies. We demonstrated the feasibility of our approach by implementing *InforShare* prototype system enabling the patient to control the sharing of his/her medical information. The logical resource model and policy schema could be eventually integrated in the formal RAMARS model providing fine-grained control for selective sharing of distributed composite resources. In addition the case study would lead to more vigorous research and experiments in applying RAMARS to secure healthcare systems. In Chapter 8, we summarize the research work described in this dissertation and discuss the potential challenging research areas we would explore in the near future.

(a) InfoShare System Portal

(b) Consent Editor

Figure 7.5: InfoShare: Patient-centric EHR Sharing

CHAPTER 8: CONCLUDING REMARKS

8.1 Summary

In this research, we proposed RAMARS framework for assured information sharing in ad-hoc collaboration. This work is particularly motivated by the desire to securely share information in spontaneous and dynamic collaboration among all distributed institutes and users. We articulated our research problems and proposed solutions in a systematic approach. Figure 8.1 summarizes the complete structure and research components that we proposed for the identified challenges.

Inside ad-hoc collaborations, a fundamental question for assured information sharing is to answer how distributed institutes and users could share distribute information resources in an authorized manner. In order to answer the question from security engineering perspective, we first clearly defined the ad-hoc collaboration environment and analyzed the generic sharing patterns for the information sharing within the environment. Instead of directly dealing with the ever-changing collaboration scopes and structures, we chose a relatively static resource-centric approach to analyzing the relationships towards the resource being shared and protected within ad-hoc collaboration, from which we addressed the fundamental need for a resource originator to be the ultimate authority over the resource and be responsible for maintaining access rights for the resource dissemination. We further derived a concept of *virtual collaborative sharing control domain* as the target domain for an originator to regulate the information sharing activities and control the information dissemination flow in distributed environments. Thereafter, we articulated the access control and trust management requirements and identified "what" security needs to be enforced.

To tackle "how" the security is enforced, we divided these identified requirements into

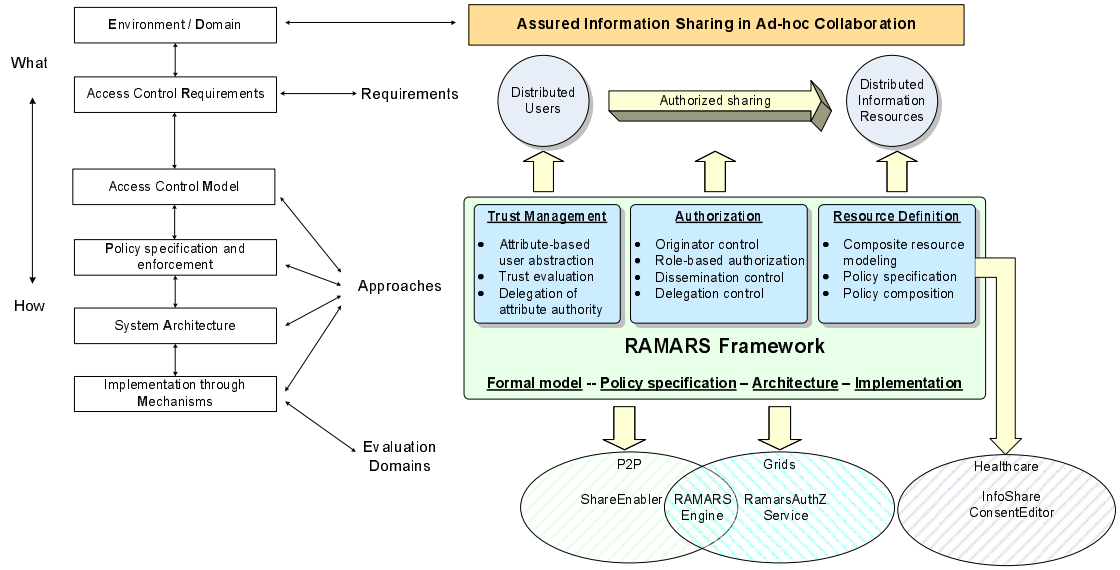


Figure 8.1: Proposed research tasks

three interrelated components and proposed solutions to each component: *trust management*, *authorization* and *resource definition*. Authorization plays a central role in bridging and enabling distributed users towards the shared resources. We proposed the originator control as the first element of authorization ensuring the originator to be the authorization authority within the virtual collaborative sharing control domain of its respective resource. We proposed a flexible role-based approach for an originator to effectively and efficiently define its collaborative sharing domain as a set of collaborator roles, and delegate various sharing capabilities to the roles. Through a set of normative sharing roles, information dissemination flow could be well regulated between the common collaborators and designated disseminators.

As an important enabling component for our role-based approach, the distributed and unknown users must be properly "identified" to be assigned the collaborator roles for sharing the originator's resources. Therefore, trust management became another component in our solution. In particular, attribute-based user classification is an effective means to abstract a large amount of distributed users. In the shadow of role-based authorization, distributed users are assigned to collaborator roles based on their attributes. Different from

traditional attribute-based access control approaches, we do not trust the user attributes by default. Instead, the evaluation of the trustworthiness of a user's attributes formulates another layer of originator control, while the delegation of attribute authority serves as an essential deciding factor in the trust evaluation process.

In terms of the distributed resources, the resource definition is the key component to enable fine-grained selective sharing, as we shifted the focus to healthcare application domain for sharing a more specific resource – Electronic Health Records (EHRs). We proposed an innovative approach to modeling the semantics and structure of EHRs, so that RAMARS policies can be applied to specific portions of the EHR structure taking into consideration critical privacy protection concerns of a patient.

These three salient components are closely coupled since they are complimentary to each other to build a comprehensive and integrated solution for assured information sharing in ad-hoc collaboration. They are seamlessly realized in our proposed RAMARS framework, with detailed formulation of access control model, policy specification, system architecture and implementation mechanisms. A P2P-based file sharing system *ShareEnabler* and an integrated Grid authorization system *RamarsAuthZ* were presented as proof-of-concept prototypes of our proposed approach. Both systems facilitate a fully distributed approach on both authorization policies and policy enforcement mechanisms, while the adoption of standardized XACML policies could achieve consistent policy interpretation and decision making across domains. The performance evaluation and system improvement demonstrated the scalability and efficiency of our RAMARS systems. As a case study of possible applications of RAMARS framework, a prototype *InfoShare* system was implemented using e-Consent mechanism to enable the patient-centric medical information sharing with different parties in the healthcare environment. The e-Consent mechanism used in our *InfoShare* system simultaneously reflects the concept of originator-control for distributed resources in the healthcare application domain.

The results of this research have potential impacts to the community in several ways.

As a general access control model, RAMARS has significant impacts to broaden RBAC's applicability into open and distributed environments. As a concrete authorization solution for ad-hoc collaboration, the ShareEnabler and RamarsAuthZ systems contribute to both P2P and Grid computing communities with first-hand practical experience for implementing advanced security services. Finally, our research practice provides a privacy protection model that could be adopted in larger electronic communication infrastructure, such as Nationwide Health Information Network (NHIN), to allow patients, physicians, hospitals, public health agencies, and other authorized users to share clinical information in a real-time and authorized manner.

8.2 Future Work

This section outlines possible future research directions based on the research work described in this dissertation.

8.2.1 Security and Compliance Analysis for Policies

Safety is a fundamental problem of access control models. In RAMARS, both the permission distribution and information dissemination flow are determined by an originator's policies. The granting of a permission or the data dissemination may consequently change the configuration of the system, and this, in turn, may introduce other permissions or information dissemination flows. The dynamic property of a distributed environment (i.e., ad-hoc collaboration environment) and distribution of trust authorities make it difficult for an authorizing authority (i.e., an originator) to foresee any possibility of permission leakage at a certain system state. Therefore, a natural security concern is whether the authorizing authority can guarantee the resource is shared legitimately. This is referred to as the safety problem in access control models. In a RAMARS system, the safety question asks, from an initial state of the system, whether or not a user can obtain a permission to access the originator's resource after a sequence of enforced policies – in other words, indirectly from certain legitimate disseminators or by updating attributes. Studying the safety and decid-

ability properties of RAMARS policies can give higher assurance for RAMARS security framework.

In addition, RAMARS systems specify and enforce XACML-based access control policies. XACML is intentionally designed to be generic and it provides great flexibility in describing access control policies. However, the flexibility and expressiveness provided by XACML come at the cost of complexity and verbosity. Especially, properties in RAMARS model may not be satisfied when these policies are specified in XACML, causing the discrepancy between what an originator intends to specify and what the actually specified XACML policies reflect. Therefore, conducting conformance checking of access control policies specified in XACML is extremely important to assure the RAMARS model being correctly enforced in RAMARS systems.

8.2.2 Trusted Computing for Originator Control

The essence of originator control is to empower an originator with ultimate control over its respective resource during the entire sharing process, within and out of the originator's administrative domain. The proposed policy-driven approach in RAMARS must be accompanied by an effective and trusted enforcement platform as the originator's policies are often enforced on the receiving client or a third-party service. As enlightened in our ShareEnabler system, on the one hand, the originator's policies and the secret key that is used to encrypt and decrypt the SEFile during distribution have to be well protected and only available to target ShareEnabler platforms. On the other hand, the target ShareEnabler platform has to be trusted not to release the file inappropriately, either by incorrect configuration or compromised software. It is commonly recognized that software alone does not provide an adequate foundation for building such a high-assurance trusted platform, since software is vulnerable to various forms of attacks such as Trojan Horses and malware. Malicious software not only can illegally read or modify sensitive data in persistent storages and memories, but also change the request for information and initiate communications to send sensitive data to other computers, which is totally unacceptable for the intention of

originator control. Fortunately, the emergence of some industry-standard trusted computing (TC) technologies [80, 104], such as Trusted Platform Module (TPM), promises a potential solution by providing hardware-based root of trust upon which secure applications can be developed. For instance, a TPM is a hardware component attached on the motherboard that never releases the root key required to access the sensitive data outside the TPM. It also provides mechanism of integrity measurement and reporting, from which strong protection capabilities and remote attestations can be achieved by an originator. Therefore, an innovative architecture with trusted computing should be investigated and integrated with current RAMARS system to guarantee an originator's authorization policies to be enforced for building high assurance systems.

8.2.3 Issues in Selective Health Information Sharing

The current research on secure and selective sharing of EHRs is just a starting point and there are still many unsolved security issues to apply RAMARS framework in building secure healthcare information sharing systems.

The concept of originator control in medical information sharing is realized in a patient-centric approach, where the patient plays a central role to control the sharing of his/her sensitive medical records among various parties. A critical prerequisite is how easily a common patient can maintain his/her access control and privacy preferences for such a huge amount of sensitive and complex information across sites, while still making the information highly usable for healthcare professionals. Therefore, a variety of analytical and empirical methods from the area of usability engineering can be also studied to verify and optimize usability of our systems.

In addition, to achieve high quality of treatment, a patient may wish to delegate the capability to grant consents to nominated representatives or medial practitioners, who may further wish to delegate the power to consent to other health professionals. Meanwhile, the patient should still maintain his control power on his medical data with proper privacy protections. Therefore, practical consent delegation and control mechanisms are crucial while

ensuring the patient's control power on his medical data with proper privacy protections.

Finally, as our approach is complementary to other existing security solutions (i.e., RBAC [15, 39] and situation-based access control [93]) for providing a fine-grained access control in healthcare systems. More in-depth research is necessary to investigate how to adapt our approach to providing a broader range of privacy preserving medical information sharing for secure healthcare services.

BIBLIOGRAPHY

[1] GT 4.0: Data Replication Service (DRS). http://www.globus.org/toolkit/docs/4.0/techpreview/datarep/.

[2] Jaxe XML Editor. http://jaxe.sourceforge.net/.

[3] Liberty Alliance. http://www.projectliberty.org/.

[4] ISO14977. Information Technology - Syntactic Metalanguage - Extended BNF, 1996.

[5] M. D. Abrams, J. Heaney, O. King, L. J. LaPadula, M. Lazear, and I. M. Ol. Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy. In *Proceedings of the 14th National Computing Security Conference*, pages 257–266, 1991.

[6] C. Adams and S. Lloyd. *Understanding Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations*. Sams Publishing, 1999.

[7] D. Agarwal and K. Berket. Supporting Dynamic Ad hoc Collaboration Capabilities. In *Proceedings of the 2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03)*, July 2003.

[8] D. Agarwal, O. Chevassut, M. R. Thompson, and G. Tsudik. An Integrated Solution for Secure Group Communication in Wide-Area Networks. In *Proceedings of the 6th IEEE Symposium on Computers and Communications*, pages 22–28, July 2001.

[9] I. Agudo, J. Lopez, and J. A. Montenegro. Attribute Delegation Based on Ontologies and Context Information. In *Proceedings of 10th IFIP TC-6 and TC-11 Conference on Communications and Multimedia Security (CMS'06), LNCS 4237*, pages 54–66. Springer, 2006.

[10] G.-J. Ahn and B. Mohan. Secure Information Sharing Using Role-based Delegation. *Journal of Network and Computer Applications*, 2, 2005.

[11] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Gianoli, F. Spataro, F. Bonnassieux, P. Broadfoot, G. Lowe, L. Cornwall, J. Jensen, D. Kelsey, A. Frohner, D. Groep, W. S. de Cerff, M. Steenbakkers, G. Venekamp, D. Kouril, A. McNab, O. Mulmo, M. Silander, J. Hahkala, and K. Lhorentey. Managing Dynamic User Communities in a Grid of Autonomous Resources. In *Proceedings of Computing in High Energy and Nuclear Physics (CHEP03)*, 2003.

[12] G. Alliance. Globus Toolkit. http://www.globus.org/toolkit/.

[13] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, N. P. C. Hong, B. Collins, N. Hardman, A. C. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N. W. Paton, D. Pearson, T. Sugden, and P. W. and. The design and implementation of Grid database services in OGSA-DAI. *Concurrency and Computation: Practice and Experience*, 17:357–376, 2005.

[14] E. Barka and R. Sandhu. Framework for Role-Based Delegation Models. In *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC)*, page 168. IEEE Computer Society, 2000.

[15] J. Barkley and K. Beznosov. Supporting relationships in access control using role based access control. In *Proceedings of 4th ACM Workshop on Role-Based Access Control*, pages 55–65, 1999.

[16] D. Bell and L. LaPadula. Secure computer systems: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306, The Mitre Corporation, Bedford, MA, March 1975.

[17] K. Berket and D. Agarwal. Enabling Secure Ad-hoc Collaboration. In *Proceedings of the Workshop on Advanced Collaborative Environments*, June 2003.

[18] K. Berket, A. Essiari, and A. Muratas. PKI-Based Security for Peer-to-Peer Information Sharing. In *Proceedings of the Fourth IEEE International Conference on Peer-to-Peer Computing*, pages 45–52, August 2004.

[19] F. Berman, G. Fox, and A. J. Hey, editors. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, May 2003. ISBN: 978-0-470-85319-1.

[20] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Specifying and enforcing access control policies for XML document sources. *World Wide Web*, 3(3):139–151, 2000.

[21] R. Bhatti, K. Moidu, and A. Ghafoor. Policy-based security management for federated healthcare databases (or RHIOs). In *Proceedings of the international workshop on Healthcare information and knowledge management*, pages 41–48, 2006.

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

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

[24] J.-W. Byun, E. Bertino, and N. Li. Purpose based access control of complex data for privacy protection. In *Proceedings of 10th ACM symposium on Access control models and technologies (SACMAT)*, pages 102–110, 2005.

[25] S. Cantor. Shibboleth Architecture, Protocols and Profiles. http://shibboleth.internet2.edu/docs/internet2-mace-shibboleth-arch-protocols-latest.pdf, September 2005.

[26] D. W. Chadwick and D. Mundy. Policy Based Electronic Transmission of Prescriptions. In *Proceedings of the 4th International Workshop on Policyies for Distributed Systems and Networks (POLICY'03)*, pages 197–206, 2003.

[27] D. W. Chadwick and A. Otenko. The PERMIS X.509 role based privilege management infrastructure. In *Proceedings of the 7th ACM symposium on Access control models and technologies (SACMAT)*, pages 135–140, 2002.

[28] D. W. Chadwick and A. Otenko. The PERMIS X.509 role based privilege management infrastructure. *Future Generation Computer Systems*, 19(2):277–289, 2003.

[29] A. Chervenak, R. Schuler, C. Kesselman, S. Koranda, and B. Moe. Wide Area Data Replication for Scientific Collaborations. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, pages 1–8, 2005.

[30] E. Coiera and R. Clarke. e-Consent: the design and implementation of consumer consent mechanisms in an electronic environment. *Journal of the American Medical Informatics Association*, 11(2):129–140, 2004.

[31] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for XML documents. *ACM Transactions on Information and System Security (TISSEC)*, 5(5):169–202, 2002.

[32] D. Denning. *Cryptography and Data Security*. Addison-Wesley, Reading, MA, 1981.

[33] Department of Defense (DoD). Trusted Computer System Evaluation Criteria (TCSEC). http://nsi.org/Library/Compsec/orangebo.txt, 1985.

[34] M. Deutsch. *Cooperation and trust: Some theoretical notes*, chapter Nebraska Symposium on Motivation, pages 275–319. University of Nebraska Press, 1962.

[35] L. L. Dimitropoulos. Privacy and Security Solutions for Interoperable Health Information Exchange: Interim Assessment of Variation Executive Summary. http://www.rti.org/pubs/avas_execsumm.pdf, July 2007. RTI Project Number 0209825.000.009.

[36] R. H. Dolin, L. Alschuler, S. Boyer, C. Beebe, F. M. Behlen, and P. V. Biron. HL7 Clinical Document Architecture, Release 2.0. ANSI Standard, 2004.

[37] D.W.Chadwick, A. Novikov, and A. Otenko. GridShib and PERMIS Integration. *Campus-Wide Information Systems*, 23(4):297–308, October 2006.

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

[39] D. M. Eyers, J. Bacon, and K. Moody. OASIS role-based access control for electronic health records. In *IEEE Proceedings – Software*, pages 16–23, 2006.

[40] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. RFC3281, http://tools.ietf.org/html/rfc3281, 2002.

[41] R. J. Feiertag, K. N. Levitt, and L. Robinson. Proving multilevel security of a system design. *ACM SIGOPS Operating Systems Review*, 11(5):57–65, 1977.

[42] Fermi National Accelerator Laboratory. The DZero Experiment. http://www-d0.fnal.gov/.

[43] E. B. Fernández, E. Gudes, and H. Song. A model for evaluation and administration of security in object-oriented databases. *IEEE Transactions on Knowledge and Data Engineering*, 6(2):275–292, 1994.

[44] D. Ferraiolo, R. Sandhu, S. Gavrila, and R. R. Kuhn. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4:224–274, August 2001.

[45] I. Foster and A. Iamnitchi. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In *Proceedings of the 2 nd International Workshop on Peer-to-Peer Systems (IPTP3'03)*, pages 118–128, 2003.

[46] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. In *Open Grid Service Infrastructure WG, Global Grid Forum*, June 2002.

[47] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *Proceedings of the 5th ACM conference on Computer and communications security*, pages 83–92, 1998.

[48] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Lecture Notes in Computer Science*, 2150, 2001.

[49] S. Franklin. XML Parsers: DOM and SAX Put to the Test. http://www.devx.com/xml/Article/16922/0/page/1, 2001.

[50] A. Gabillon and E. Bruno. Regulating access to XML documents. In *Proceedings of the 15th annual working conference on Database and application security*, pages 299–314, 2001.

[51] D. Gambetta, editor. *Trust: Making and Breaking Cooperative Relations*. Blackwell Publishers, 1990.

[52] Globus. GT 4.0 Data Management. http://www.globus.org/toolkit/docs/4.0/data/.

[53] The Globus Alliance. http://www.globus.org.

[54] Globus. Towards Open Grid Services Architecture. http://www.globus.org/ogsa/.

[55] GridShib Project. GridShib: a Policy Controlled Attribute Framework. http://gridshib.globus.org/.

[56] R. Groeper, C. Grimm, S. Piger, and J. Wiebelitz. An Architecture for Authorization in Grids using Shibboleth and VOMS. In *Proceedings of 33rd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2007)*, pages 367–374, 2007.

[57] HL7. Health Level 7 (HL7). http://www.hl7.org.

[58] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC3280, http://rfc.net/rfc3280.html, 2002.

[59] IAIK. IAIK Crypto Toolkits for the Java platform. http://jce.iaik.tugraz.at/.

[60] IEEE-USA's Medical Technology Policy Committee Interoperability Working Group, editor. *Interoperability for the National Health Information Network (NHIN)*. IEEE-USA EBOOKS, 2006.

[61] The TLS Protocol Version 1.0. http://www.ietf.org/frc/rfc2246.txt.

[62] Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names. http://www.faqs.org/rfcs/rfc2253.html.

[63] Iowa Foundation for Medical Care. HISPC State Implementation Project Summary and Impact Analysis Report for the State of Iowa. http://www.ifmc.org/news/State Impact Report_11-27-07.doc, 2007.

[64] ITU-T Rec. X.509 ISO/IEC 9594-8. The Directory: Public-Key and Attribute Certificate Frameworks, May 2001.

[65] T. Jim. SD3: a trust management system with certificate. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 106–115. IEEE Computer Society Press, 2001.

[66] J. Jin and G.-J. Ahn. Role-based Access Management for Ad-hoc Collaborative Sharing. In *Proceedings of 11th Symposium on Access Control Models and Technologies (SACMAT)*, pages 200–209, 2006.

[67] J. Jin and G.-J. Ahn. ShareEnabler: Policy-Driven Access Management for Ad-hoc Collaborative Sharing. In *Proceedings of 2nd International Workshop on Pervasive Information Management (PIM 2006), Lecture Notes in Computer Science (LNCS-4254)*, pages 724–740, 2006.

[68] 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*, pages 41–48, November 2007.

[69] K.Berket, D.A.Agarwal, and O. Chevassut. A Practical Approach to the InterGroup Protocols. *Future Generation Computer Systems*, 18(5):709–719, 2002.

[70] B. W. Lampson. Protection. In *Proceedings of 5th Princeton Conference on Information Sciences and Systems*, pages 437–443. Princeton, 1971. Reprinted in ACM Operating Systems Rev. 8, 1, pages 18-24 (Jan. 1974).

[71] B. Lang, I. Foster, F. Siebenlist, R. Ananthakrishnan, and T. Freeman. A Flexible Attribute Based Access Control Method for Grid Computing. *Journal of Grid Computing*, 7(2), 2008.

[72] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation Logic: A logic-based approach to distributed authorization. *ACM Transaction on Information and System Security (TISSEC)*, 6(1):128–171, 2003.

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

[74] J. Linn and M. Nystrom. Attribute certification: An enabling technology for delegation and role-based controls in distributed environments. In *Proceedings of 4th ACM Workshop on Role-Based Access Control*, pages 121–130, Fairfax, VA, October 1999.

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

[76] LionShare: Connecting and Extending Peer-to-Peer Networks (LionShare White Paper). http://lionshare.its.psu.edu/main/info/docspresentation/ LionShareWP.pdf, October 2004.

[77] 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*, pages 265–276, 2008.

[78] E. Maler, P. Mishra, and R. Philpott. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML). Oasis Standard, September 2003.

[79] C. J. McCollum, J. R. Messing, and L. Notargiacomo. Beyond the pale of MAC and DAC — defining new forms of access control. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 190–200, 1990.

[80] C. Mitchell. *Trusted Computing (Professional Applications of Computing)*. IEEE Press, 2005.

[81] A. Nadalin, C. Kaler, R. Monzillo, and P. Hallam-Baker. Web Services Security: SOAP Message Security 1.1. OASIS Standard Specification, February 2006.

[82] OASIS. Security Assertion Markup Language. http://www.oasis-open.org/committees/security/.

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

[84] OASIS. XACML 2.0 Core: eXtensible Access Control Markup Language (XACML) Version 2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, February 2005.

[85] OASIS. XACML 3.0 administrative policy Working draft 10, December 2005.

[86] C. M. O'Keefe, P. Greenfield, and A. Goodchild. A Decentralised Approach to Electronic Consent and Health Information Access Control. *Journal of Research and Practice in Information Technology*, 37(2):161–178, 2005.

[87] openEHR Community. openEHR. http://www.openehr.org.

[88] A. Oram. *Peer-to-peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, 2001.

[89] S. Osborn, R. Sandhu, and Q. Munawer. Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(2):85–106, 2000.

[90] J. Park and R. Sandhu. Originator Control in Usage Control. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, pages 60–66, June 2002.

[91] J. Park and R. Sandhu. Towards usage control models: beyond traditional access control. In *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, pages 57–64, 2002.

[92] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A Community Authorization Service for Group Collaboration. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, pages 50–59, June 2002.

[93] M. Peleg, D. Beimel, D. Dori, and Y. Denekamp. Situation-Based Access Control: Privacy management via modeling of patient data access scenarios. *Journal of Biomedical Informatics*, 41(6):1028–1040, 2008.

[94] J. Pritts and K. Connor. The Implementation of E-Consent Mechanisms in Three Countries: Canada, England, and the Netherlands. SAMHSA report, http://ihcrp.georgetown.edu/pdfs/prittse-consent.pdf, 2007.

[95] F. Rabitti, E. Bertino, W. Kim, and D. Woelk. A model of authorization for next-generation database systems. *ACM Transactions on Database Systems (TODS)*, 16(1):88–131, 1991.

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

[97] A. Rajasekar, M. Wan, and R. Moore. MySRB and SRB – Components of a Data Grid. In *Proceedings of International Symposium on High Performance Distributed Computing (HPDC)*, pages 301–310, 2002.

[98] Uniform Resource Identifiers (URI): Generic Syntax. http://rfc.net/rfc2396.html.

[99] R. L. Rivest and B. Lampson. SDSI: A Simple Distributed Security Infrastructure. http://research.microsoft.com/lampson/59-SDSI/WebPage.html, 1996.

[100] C. Ruan and V. Varadharajan. An Authorization Model for E-consent Requirement in a Health Care Application. *Applied Cryptography and Network Security, LNCS*, 2846:191–205, 2003.

[101] R. Sandhu. Engineering authority and trust in cyberspace: the OM-AM and RBAC way. In *Proceedings of the 5th ACM Workshop on Role-based Access Control*, pages 111–119, 2000.

[102] R. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role Based Access Control Models. *IEEE Computer*, 29, February 1996.

[103] R. Sandhu and P. Samarati. Access Control: Principles and Practice. *IEEE Communications*, 32(9):40–48, September 1994.

[104] R. Sandhu and X. Zhang. Peer-to-peer access control architecture using trusted computing technology. In *Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 147–158, 2005.

[105] H. Shen and P. Dewan. Access control for collaborative environments. In *Proceedings of ACM conference on Computer-supported cooperative work (CSCW)*, pages 51–58, 1992.

[106] Swedish Institute of Computer Science. XACML 3.0 Administrative Policy support (Beta version). http://www.sics.se/spot/xacml_3_0.html, 2006.

[107] R. Thomas and R. Sandhu. Towards a Multi-dimensional Characterization of Dissemination Control. In *Proceedings of the 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'04)*, pages 197–200, 2004.

[108] M. Thompson, A. Essiari, and S. Mudumbai. Certificate-based Authorization Policy in a PKI Environment. *ACM Transaction on Information and System Security (TISSEC)*, 6(4):566–588, November 2003.

[109] M. Thompson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essiari. Certificate-based access control for widely distributed resources. In *Proceedings of 8th USENIX Security Symposium*, pages 23–26, 1999.

[110] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC 3820, 2004.

[111] Waste. http://waste.sourceforge.net/.

[112] V. Welch, R. Ananthakrishnan, F. Siebenlist, D. Chadwick, S. Meder, and L. Pearlman. Use of SAML for OGSI Authorization. https://forge.gridforum.org/projects/ogsa-authz/document/draft-ogsi-authz-saml-aug15-05.pdf/en/1, August 2005.

[113] V. Welch, F. Siebenlist, I. Foster, J. Bresnahan, K. Czajkowski, J. Gawor, C. Kesselman, S. M. L. Pearlman, and S. Tuecke. Security for Grid Services. In *Proceedings of 12th IEEE International Symposium on High Performance Distributed Computing*, pages 48–57, 2003.

[114] N. Yang, H. Barringer, and N. Zhang. A Purpose-Based Access Control Model. In *Proceedings of 3rd International Symposium on Information Assurance and Security (IAS)*, pages 143–148, 2007.

[115] J. P. Yoon. Presto authorization: a bitmap indexing scheme for high-speed access control to XML documents. *IEEE Transactions on Knowledge and Data Engineering*, 18(7):971– 987, July 2006.

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