A Role-Based Delegation Framework for Healthcare Information Systems

Longhua Zhang
College of Information Technology,
UNC Charlotte
Charlotte, NC 28223, USA
lozhang@uncc.edu

Gail-Joon Ahn
Laboratory of Information Integration, Security and Privacy (LIISP)
Dept. of Software & Information Systems,
UNC Charlotte
Charlotte, NC 28223, USA
gahn@uncc.edu

Bei-Tseng Chu
Laboratory of Information Integration, Security and Privacy (LIISP)
Dept. of Software & Information Systems, UNC Charlotte
Charlotte, NC 28223, USA
billchu@uncc.edu

ABSTRACT
As organizations implement information strategies that call for sharing access to resources in the networked environment, mechanisms must be provided to protect the resources from adversaries. The proposed delegation framework addresses the issue of how to advocate selective information sharing in role-based systems while minimizing the risks of unauthorized access. We introduce a systematic approach to specify delegation and revocation policies using a set of rules. We demonstrate the feasibility of our framework through policy specification, enforcement, and a proof-of-concept implementation on specific domains, e.g. the healthcare environment. We believe that our work can be applied to organizations that rely heavily on collaborative tasks.

Categories and Subject Descriptors
K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms
Management, Security

Keywords
Role, Access Control, Delegation, Revocation, Healthcare Information System.

1. INTRODUCTION
The healthcare industry is as competitive and multifaceted as any industry in the world today. The healthcare information system provides many advantages when used for improved access, collaboration and data sharing among healthcare providers, patients, and researchers. Yet considering the highly personal and potentially destructive nature of the medical data, it comes with significant risks to the confidentiality, integrity, and availability of such information. Currently, most healthcare information systems have supported minimal security features: data transmission may be encrypted; passwords, public and private keys are used to provide protection from adversaries. The problem that remains, and addressed here, is how to enable selective information sharing without the risk of exposing additional information that needs to be protected.

In [1], Wiederhold et al. proposed a centralized solution to assign a security officer the responsibility to manage sharing of sensitive information. They formalized the role of a security officer who has responsibility to assure that no appropriate information can leave an enterprise domain. But under the healthcare environment, the information sharing tends to be very dynamic and often ad hoc. Hence, this centralized management approach is not appropriate to the healthcare domain because the workload on such an officer (or a small group of security officers) will be overwhelming. Since the very goal of our research is to enable users to access and selectively share resources in distributed systems, we assume that users can be trusted to exercise their discretions on resources. If Alice explicitly shares a resource with Bob, she trusts Bob to use the resource. We also consider enhancing the scalability of information sharing. One promising approach is through delegation [3].

In today's distributed systems, all the resources required to carry out an operation are rarely local to the system to which the user is logged in. Delegation is more often the rule than the exception. There are many definitions and different types of delegation in the literature [2, 3]. In general, delegation is referred to as one active entity in a system delegates its authority to another entity to carry out some functions on behalf of the former. Through delegation, individual user is trusted and empowered to share resources to which they have access.

In this paper, we introduce a role-based delegation framework for information sharing in healthcare information systems. Our framework adapts a role-based delegation model called RDM2000 proposed by Zhang, et al [3]. Based on that, we develop system architecture to apply RDM2000 to the healthcare information systems. We demonstrate the feasibility of our approach through a proof-of-concept prototype implementation.

SACMAT'02, June 3-4, 2002, Monterey, California, USA.

The rest of this paper is organized as follows. Section 2 describes background and related works. Section 3 presents the delegation framework for the healthcare setting. In section 4, we describe the architecture and implementation of the proposed framework. Section 5 concludes this paper and outlines our future work.

2. BACKGROUND AND RELATED WORK

Delegation requirements arise when a user may need to act on another user's behalf for accessing resources. There are many definitions of delegation in the literature. In general, it is referred to as the process whereby one active entity in a system authorizes another entity to act on behalf of the former by transferring a set of rights [2, 3]. We first address some examples in the healthcare setting to clarify the problem, and then give an overview of related work.

2.1 Background

In a healthcare organization, a wide variety of information on its patients is needed to provide effective medical services. The main purpose of healthcare information systems is to provide a fully integrated electronic patient record. Briefly, it includes traditional clerical information about appointments and admissions; results from specialties such as pathology, radiology, and endoscopy; drug treatment; procedures; and problem lists. In addition, it generates and stores plans for nursing care, clinical correspondence, and dictated note from ward rounds.

During a simple healthcare episode, many professionals involve in a number of medical acts. Healthcare administration personnel, healthcare professionals, social care professionals, as well as patients need to selectively interact with the healthcare information. The specific level of access and permissions a user can have to the healthcare information will be determined by his responsibilities in the organization. In order to achieve this, users are identified to the system as having one or more roles, such as ward base nurse, specialist nurse, junior doctor, ward clerk, clinical consultant, neurologist, gynecologist, radiologist, etc. Only a specialist doctor may be allowed to see a section of the records of his patient that pertain to the results of very sensitive medical test. However, in some situations, a specialist doctor may need to share information with other specialists within or across organizational boundaries. Consider the case of a virtual hospital that consists of several highly collaborative healthcare organizations connected by high-speed network, as shown in figure 1. In this example, *Jennifer* is under the care of a Neurologist, Dr. *Chen*. Suppose *Jennifer* becomes pregnant and becomes a patient of Dr. *Jain*, a Gynecologist. Dr. *Chen* and Dr. *Jain* must collaborate very closely to share information during *Jennifer*'s pregnancy. Dr. Chen may further consult Dr. *White* in a specialist clinic to prescribe a drug for *Jennifer*. Thus Dr. *White* needs access to *Jennifer*'s records too.

Another example we use to motivate our discussions is a hospital's policy to enable access to anonymous medical data for research purposes. Medical research promotes human knowledge to improve the quality of healthcare; therefore, it should be encouraged, stimulated, and promoted as strongly as possible. However, preservation of confidentiality and respect

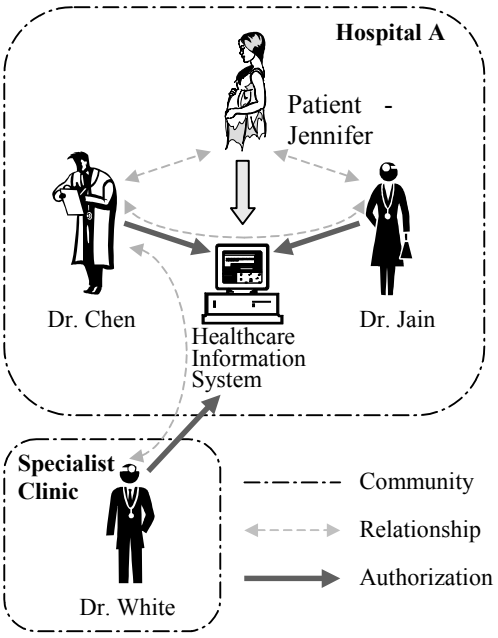

Figure 1. Information sharing in the healthcare environment

for patient's rights should take precedence over any scientific purpose. For example, anonymous medical data removes names and social security numbers from patients' records. But removing names and social security numbers doesn't ensure privacy and confidentiality of medical information. Most of the US population can be uniquely identified by combination of birth date, sex, and ZIP code. Thus, a hospital may limit the access to anonymous medical data only to authorized people, e.g. only cardiologists are allowed to access cardiac medical records.

We observe the following commonalities between two examples above. First, selective information sharing is necessary. We are dealing with friends, not enemies, and should provide relevant information expeditiously. Second, the information may be shared across organizational boundaries. Medical records may be exchanged between collaborative hospitals for shared patient; researchers may reside in different healthcare organizations. Because sharing a resource across organizational boundaries often means authorizing a server to give access to a third party, it implies enabling resource servers to reason about previously unknown third parties. This requirement contrasts with many conventional systems, wherein a server need only reason about the set of users known inside a given organization. Third, it is impossible to fully predicate what data should be shared, when and to whom. And another thing is that a mechanism must be provided for revoking the sharing when it is no longer needed. All these factors have to be considered in order to formulate the mechanism for information sharing in healthcare organizations.

2.2 Related Work

Historically, the access control problem has been couched based on subjects and objects [5]. The subjects may be users or processes acting on behalf of users. The objects are data or

resources in the system. Permissions are a set of operations that a subject can have with one or more objects in the system. Over the last few decades, we have seen the evolution and development of many access control models [4, 5]. As organizations implement information strategies that call for sharing access to resources in the networked environment, access control concerns not only the protection of individual objects and subjects, but also the management of access control decisions in dynamic, highly distributed systems. Various approaches have been proposed.

Thomas et. Al formulated team-based access control (TMAC) [18] and task-based access control (TBAC) [5] as active security models. This approach models access control from a context-oriented perspective than the traditional subject-object one. TMAC and TBAC are aware of the context information associated with an ongoing activity. Thus, they provide a natural way to control access for collaborative activities in teams and workflows. However, We argue that TBAC modes are specific configurations of role-based access control, where context information can be viewed as constraints.

Role-based access control is an enabling technology for managing and enforcing security in large-scale and enterprise-wide systems. The basic notion of RBAC is that permissions are associated with roles, users are assigned to appropriate roles, and users acquire permissions by being members of roles. Users can be easily reassigned from one role to another. Roles can be granted new permissions. And permissions can be easily revoked from roles as needed. This greatly simplifies security management [4]. Constraints can apply to relations and functions defined in an RBAC model to establish higher-level organizational policy.

Delegation is another important factor for secure distributed computing environment [3]. In large role-based systems, the number of roles may be in the hundreds or thousands, and users in the tens or hundreds of thousands. In addition, today's dynamic and collaborative work environment may require users assuming temporary roles. Management of user assignment is a formidable task and could not realistically be centralized to a small group of security officers. Decentralizing administration of user assignment is critical in distributed role-based access control. It is natural to decentralize the administration through delegation to increase the scalability of role-based systems. The basic idea behind a role-based delegation is that users themselves may delegate role authorities to other users to carry out some functions authorized to the former.

Several papers have been published on security requirements in healthcare environment [13, 14]. Projects have been undertaken to explore the use of RBAC and identify sample RBAC policies in healthcare information systems [12, 15]. It is generally accepted that RBAC is more suited to healthcare than other access control mechanisms to meet the requirements for the security of healthcare information. Also, we need to consider the delegation needs for efficient collaborative environment. The purpose of this paper is to investigate how to enhance the information sharing in healthcare information system through role-based access control and delegation.

3. ROLE-BASED DELEGATION FRAMEWORK

The Role-Based delegation framework was initially developed to provide a means of decentralizing user assignment in large distributed role-based systems. The framework includes a delegation model (RDM2000) [3] and a rule-based language for specifying and enforcing delegation and revocation policies. Since the delegation framework cannot exist without implementation of RBAC, we first give a brief discussion of applying RBAC to healthcare information systems. Throughout the design of the framework, RBAC is used as a foundation. Our framework builds upon prior works on RBAC models, extending them to incorporate the delegation and revocation notions.

3.1 RBAC in Healthcare Information System

RBAC offers an elegant solution to the problem of managing complex access control rule sets in distributed systems. The basis of RBAC is the concept of roles, which is a group mechanism used to categorize users based on various properties, such as job title, job functions, or responsibilities. Permissions are associated with roles, users are assigned to appropriate roles, and users acquire permissions by being members of roles.

Although RBAC is very useful for modeling access control in a variety of applications, traditional RBAC is difficult to capture security-relevant contexts that would have an impact on access decisions. In a healthcare setting, the specific level of access and permissions a user can have to the system will be determined not only by his role in the organization, but also the relevant security context, such as the patient, location, and time. Georgiadis et al. [9] introduced environment roles to capture security-relevant contexts in role-based system. Giuri et al [16] proposed the use of role templates and parameterized permissions to address the same problem. Our approach is similar to role templates. We consider security context as a special kind of constraints in RBAC. As we will address in the subsequent section, the idea is to assign context constraints to users first, and then apply them to role activations. The RBAC with context constraints is illustrated as shown in figure 2.

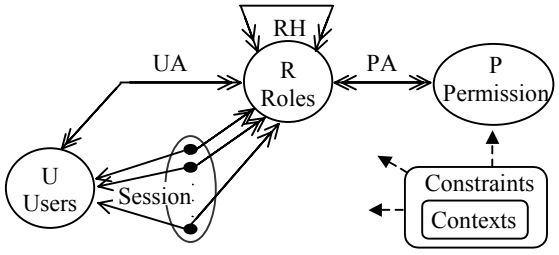

Figure 2. RBAC with context constraints

3.1.1 Users

X.509 certificates [10] are used to identify users in the framework. The basic purpose of X.509 certificates is simply the binding of users to keys. An X.509 certificate is digitally signed by the issuer of the certificate (certificate authority) that has confirmed the binding between the public-key and the holder of the certificate. Roles are assigned to user's public key in the

certificate. During the login phase, a user presents his X.509 certificate and gets authenticated. After authentication, the roles assigned to the public key are retrieved from database for the user to activate. If the healthcare information needs to be transmitted across organizational boundary, the X.509 certificate is also used to establish a secure channel for encrypted data transmission.

There are two types of users in a healthcare information system: one is internal users which consists mainly of treating personnel as well as related clerks, e.g. bill clerks; the other is external users such as researchers, consultants, and insurance companies which reside across the organizational boundary. The reason we distinguish between these users is for administration purpose, since a healthcare organization may have more strict management policies for external users.

3.1.2 Role and Role Hierarchies

One crucial construct provided by RBAC is the role and role hierarchy. The role hierarchy is a natural means for structuring roles to reflect an organization's lines of authority and responsibility. It is organized in partial order $\geq$, so that if $x \geq y$ then role x inherits the permissions of y. A member of x is also implicitly a member of y. in such case, x is said to be senior to y [4]. Role hierarchies allow a security officer to specify generic access rules just once, rather than for every role to which the rules apply. A role hierarchy example in a hospital is shown in figure 3.

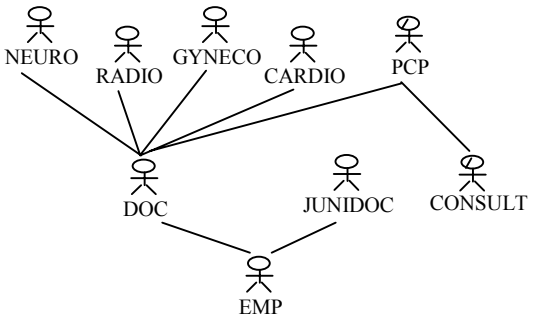

Figure 3. A role hierarchy example in a hospital

3.1.3 Permissions

In healthcare information systems, patients' medical information is saved as records in database. Objects of access control in the system are views. Views in relational databases have been considered as an ideal approach for access control of objects, because they have a higher degree of logical abstraction than physical data to enable context-based or content-based security. Thus a permission defined in RBAC represents an access method to one or more views, e.g. select, insert, update, and delete.

3.1.4 Context Constraints

The access control in a healthcare setting involves a variety of security related contexts [9]. We consider the following context variables in the proposed framework: patient and location.

- Patient: the specific patient a healthcare employee is in care of.
- Location: the specific address where the access request is originated.

We use the function *patient(u)* to return all the patients u is in care of; and *location(u)* to return current location of u.

Context constraints are assigned to users and anchored to roles. And these context constraints are applied to role activation and permission check. For example, it might be a hospital's security policy that a doctor can activate his role of ERP (Emergency Room Physician) to the information system only when he is in the emergency room. The context constraints are inherited according to role hierarchies.

3.1.5 Role Activation with Contexts

Role-based systems usually treat roles as static attributes or at least attributes that change infrequently. That is, we might not allow the following scenario: A hospital employee work as a doctor in the morning, as a billing clerk in the afternoon, and then as a doctor again the next day. As a result, roles are usually defined with a fixed set of permissions. Unfortunately, in a healthcare setting, permissions assigned to a role are not always static. Sometimes the permissions assigned to a role should be given depend on what the member of the role is currently doing, or the security-related contexts. For example, suppose a hospital's privacy policy grants access to sensitive patient information only to the patient's Primary Care Physician (PCP). What permissions should be assigned to the role of PCP? It is inappropriate to grant the permission to all patients' records to PCP. A doctor should be granted the permissions assigned to the PCP of a patient only when the patient has designated him as the PCP. We define this kind of permission role assignment as a *dynamic permission assignment*, where a set of permissions is assigned to a role at run-time. This dynamic permission assignment is achieved by applying anchored context constraints during the role activation process in a session.

Traditional RBAC permission activation has three steps. At first, a user presents suitable credentials to complete the identification and authentication procedure; then the user has to select a subset of roles from the assigned role set for activating in current session; finally, a particular set of permissions assigned to the subset of roles is granted to the user. In order to apply context constraints, we change the traditional permission activation process as follows. After successful authentication, the user selects a subset of roles for activating in current session. The anchored context constraints from the user are retrieved and applied at role activation; after successful role activation, the anchored context constraints further are applied to permissions of the activated roles; finally a particular set of permissions is granted to the user.

3.2 RDM2000 Model

RDM2000 [3] is an extension of RBAC96 [4]. The scope of RDM2000 is to address user-to-user delegation supporting role hierarchies and multi-step delegation in role-based systems. A new relation called delegation relation (*DLGT)* is defined. It includes sets of three elements: original user assignments *UAO*, delegated user assignment *UAD*, and constraints. The motivation behind this relation is to address the relationships among different components involved in a delegation. There are four components defined in a delegation relation: a delegating user, a delegating role, a delegated user, and a delegated role. A delegation relation is one-to-many relationship on user assignments. It consists of original user delegation (*ODLGT*) and delegated user delegation (*DDLGT*). Two types of delegation are introduced: *single-step delegation* and *multi-step delegation*. Single-step delegation does not allow the delegated role to be further delegated; Multi-step delegation allows multiple delegations until it reaches the maximum delegation depth. The maximum delegation depth is a natural number defined to impose restriction on the delegation. Single-step delegation is a special case of multi-step delegation with zero maximum delegation depth. A delegation path is an ordered list of user assignment relations generated through multi-step delegation. A delegation path always starts from an original user assignment. Delegation paths starting with the same original user assignment can further construct a delegation tree. A delegation tree expresses the delegation path in a hierarchical structure. Each node in the tree refers to a user assignment and each edge to a delegation relation. The RDM2000 components are depicted in figure 4. Different revocation semantics are also addressed in RDM2000.

Constraints are an important aspect of RBAC96 and RDM2000 and can lay out higher-level organizational policies. In theory, the effects of constraints can be achieved by establishing procedures and sedulous actions of security administrators [6]. In RDM2000, the constraints are enforced by a set of integrity rules that provide management and regulators with the confidence that critical security policies are uniformly and consistently enforced. In the framework, when a user delegates a role, all context constraints that are assigned to the user and anchored to the delegated role are delegated as well.

3.3 Rule-Based Policy Specification Language

RDM2000 defines policies that allow regular users to delegate their roles. It also specifies the policies regarding which delegated roles can be revoked. A rule-based language [3] is adopted to specify and enforce these policies. It is a declarative language in which binds logic with rules. The advantage is that it is entirely declarative so it is easier for security administrator to define policies.

A *rule* takes the form:

$H \leftarrow F1\&F2\&\ldots\&Fn$

where H, F1, F2, …, Fn are Boolean functions.

There are three sets of rules in the framework: basic authorization rules specify organizational delegation and revocation policies; authorization derivation rules enforce these policies in the healthcare information system; and integrity rules specify and enforce role-based constraints.

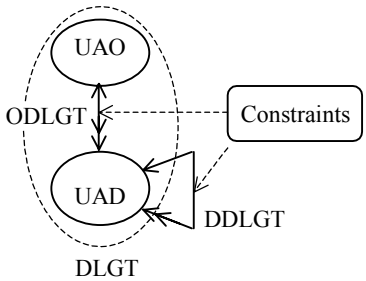

Figure 4. RDM2000

For example, a **user-user delegation authorization rule** forms as follows:

can_delegate(r, cr, n) ← .

where r, cr, and n are elements of roles, prerequisite conditions, and maximum delegation depths respectively.

This is the basic user-to-user delegation authorization rule. It means that a member of the role *r* (or a member of any role that is senior to *r*) can assign a user whose current membership satisfies prerequisite condition *cr* to role *r* (or a role that is junior to *r*) without exceeding the maximum delegation depth *n*.

A user delegation request is further authorized by the **user-user delegation authorization derivation rule** that takes the form:

der_can_delegate(u, r, u', r', dlg_opt) ←
can_delegate(r", cr, n)&
active(u, r, s)&
delegatable(u, r)&
senior(r, r")&
in(u', cr)&
junior(r', r")&
in(depth(u, r), n).

where u and u' are elements of users; r, r', and r" are elements of roles; cr and s are elements of prerequisite condition and sessions respectively; dlg_opt is a Boolean term, if it is true, then further delegation is allowed. This argument is used as Boolean control of delegation propagation.

This rule means that a user *u* with a membership of a role *r* senior to *r"* activated in session *s* can delegate a user *u'* whose current role membership satisfies prerequisite condition *cr* to role *r'* (*r'* is junior to role *r"*) without exceeding the maximum delegation depth *n*.

Similar rules [3] are defined for role-based revocations and are applied to specify constraints.

4. SYSTEM ARCHITECTURE AND IMPLEMENTATION

4.1 The Architecture

The notions described in RDM2000 and the rule-based policy specification language are designed to be utilized within an administrative-directed delegation management architecture [3]. An overview of the architecture is shown in figure 5. It consists of a number of services and management agents together with the objects to be managed. The enforcement agents are based on a combination of roles and rules for specifying and interpreting policies [8]. Since delegation and revocation services are only

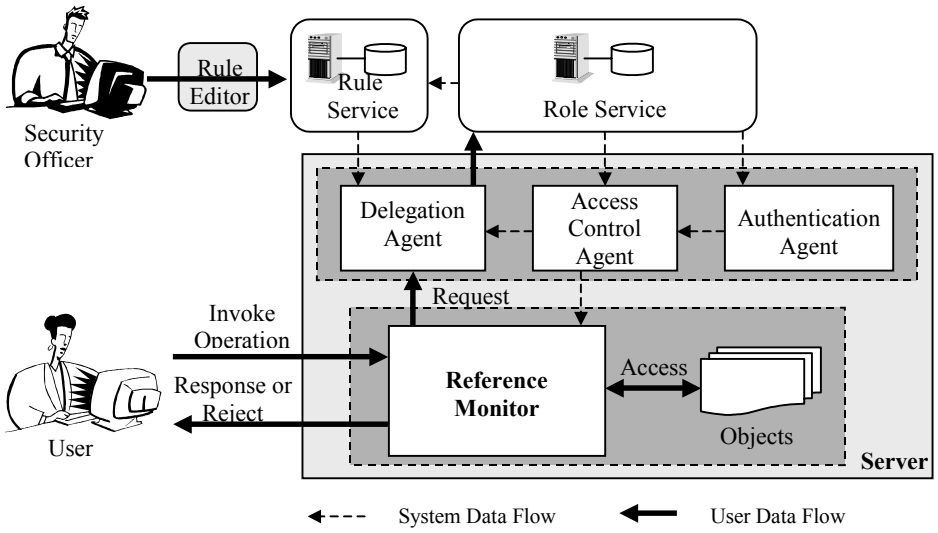

Figure 5. Overview of access control and delegation architecture

part of a security infrastructure, we choose a modular approach to our architecture that allows the delegation and revocation services to work with current and future authentication and access control services. The modularity enables future enhancements of our approach. This section briefly discusses the functionality of these main building blocks.

4.1.1 Role Service

The role service is provided by a role server, which is an implementation of the RBAC96 and RDM2000 components. A role server maintains RBAC database and provides user credentials, role memberships, associated permissions, and delegation relations of the system. These elements are created and maintained using a set of graphical administration tools. These tools can also be used to maintain the integrity of database elements by checking and enforcing integrity rules [3, 7]. In this paper, we provide administration tools for managing RDM2000 elements. The administration tools for RBAC components are beyond the scope of this paper because we can simply adopt an existing tool for that purpose such as [6, 7].

4.1.2 Rule Service

The rule service is provided by a rule server, which is used to manage delegation and revocation rules. A delegation or a revocation rule is always associated with a role, which specifies the role that can be delegated. Delegation rules are not meant to be used for the user role assignment by security officers. Also, the delegation and revocation rules do not control the actual delegation and revocation of role memberships. They are implemented as authorization policies that authorize requests from users. Rule management will be explained in more detail in section 4.2.

4.1.3 Delegation Agent

The delegation agent is an administrative infrastructure, which authorizes delegation and revocation requests from users by applying derivation authorization rules and processes delegation and revocation transactions on behalf of users. A delegation agent registers to both the rule service and the role service. It has a rule engine to optimize rule search, interpret rules and authorize user requests. The result of an authorized delegation or revocation is sent and saved to the RBAC database.

4.1.4 Authentication and Access Control

The implementation requirements related to the delegation framework are not only a delegation agent, but also authentication and access control agents. The authentication agent is used to authenticate users during their initial sign-on and supply them with an initial set of credentials. Authentication agents are registered with the role service. The reference monitor makes access control decisions based on information supplied by the access control agent.

4.2 Rule Management

Rules define the authorization policies for delegation and revocation. Rules are associated with roles, as shown in figure 6. In large role-based systems, the number of roles may be in the hundreds or thousands, the management of associated rules is a tremendous task. It is necessary to decentralize the management activities among multiple security officers and provide tools to automate the administration tasks.

4.2.1 Rule Life-Cycle

During its lifetime a rule undergoes various changes of status. During the creation phase, it acquires its dormant status. It keeps this status, undergoing editing by a security officer until it is enabled or disabled. An enabled rule can be disabled and vice versa. The disable status is used for the purpose of preserve a rule without enacting it. Also a rule can be deleted. A deleted rule does not actually exist anymore. The rule editor provides a convenient means to change the status of a rule.

4.2.2 Rule Editor

In large role-based system, there may be tens or hundreds of delegation and revocation rules. The rule editor is developed to

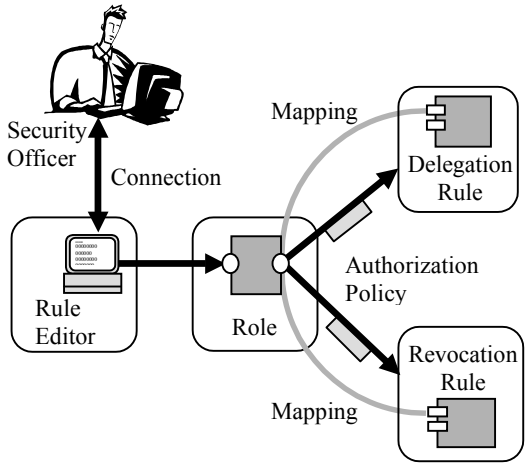

Figure 6. Rule Management

simplify the management of these rules. As a portion of an integrated RBAC administration platform developed to manage various RBAC and RDM2000 components, the rule editor is used to view, create, edit, and delete delegation and revocation rules, as shown in figure 7. The rule editor consists of two panes: the hierarchy pane is used to view the role hierarchies and associated rules in the system displaying parent-child relationships. The rule pane is used to examine and edit the attributes of a rule, including the status of the rule. Note that after selecting a rule to edit, the security officer can switch the left pane from tree-based view to graph-based role hierarchy view and define prerequisite condition.

4.3 Delegation and Revocation Walkthrough

In this section, we use the example requirements specified in section 2.1 to illustrate the delegation and revocation scenarios in the healthcare setting. We describe the message exchanged in each scenario.

The role hierarchy example shown in figure 3 is used as the role hierarchy in hospital A. Suppose the security officer in the hospital has defined the following delegation and revocation authorization rules:

Rule 1: can_delegate(NEURO, DOC, 1) ← .
Rule 2: can_revokeGD(NEURO) ← .
Rule 3: can_delegate(PCP, TRUSTED_VEMP, 1) ← .
Rule 4: can_revokeGI(NEURO) ← .
Rule 5: can_revokeGD(PCP) ← .

The first rule says a member of the role NEURO can delegate role NEURO (or a role that is junior to NEURO) to a user who is a member of DOC without exceeding the maximum delegation depth of one. The second rule says the delegated role NEURO can be grant-dependently revoked. The third rule says a member of the role PCP can delegate role PCP (or a role that is junior to PCP) to a user who is a member of TRUSTED_VEMP (trusted virtual employee) without exceeding the maximum delegation depth of one. Note that the role TRUSTED_VEMP indicates the employee status from a trusted organization. In the virtual hospital example, we assume a user is a member of role TRUSTED_VEMP if the user's X.509 certificate is signed by a trusted organization. For example, if Dr. *White*'s X.509 certificate is signed by the Specialist Clinic and the Specialist Clinic is trusted by hospital A, then Dr. *White* assumes role TRUSTED_VEMP in Hospital A. It is obvious that a member of EMP in hospital A must be a member of TRUSTED_VEMP since an organization always trusts itself. The last two rules authorize revocations.

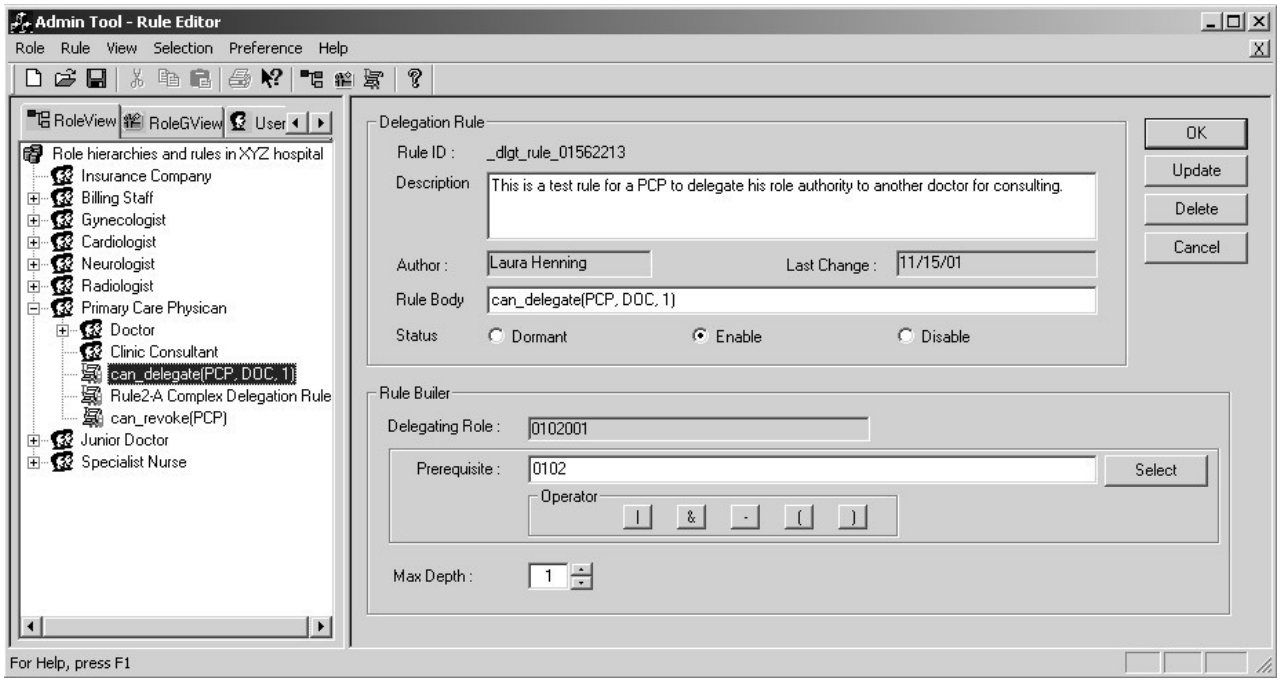

Figure 7. Rule editor

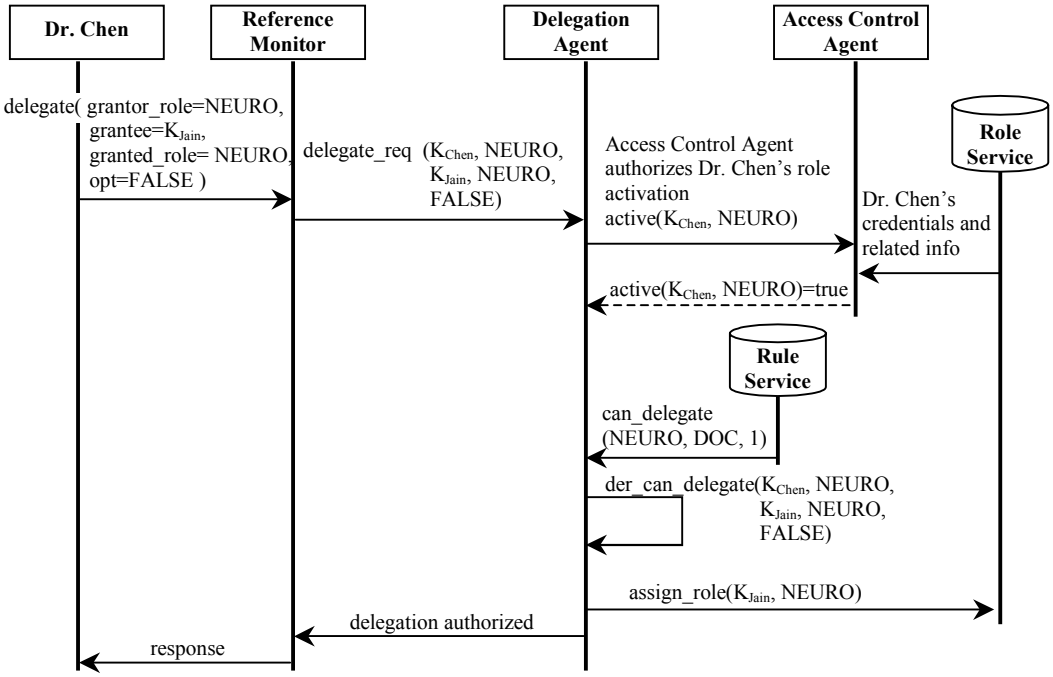

Figure 8. The sequence diagram for a delegation scenario

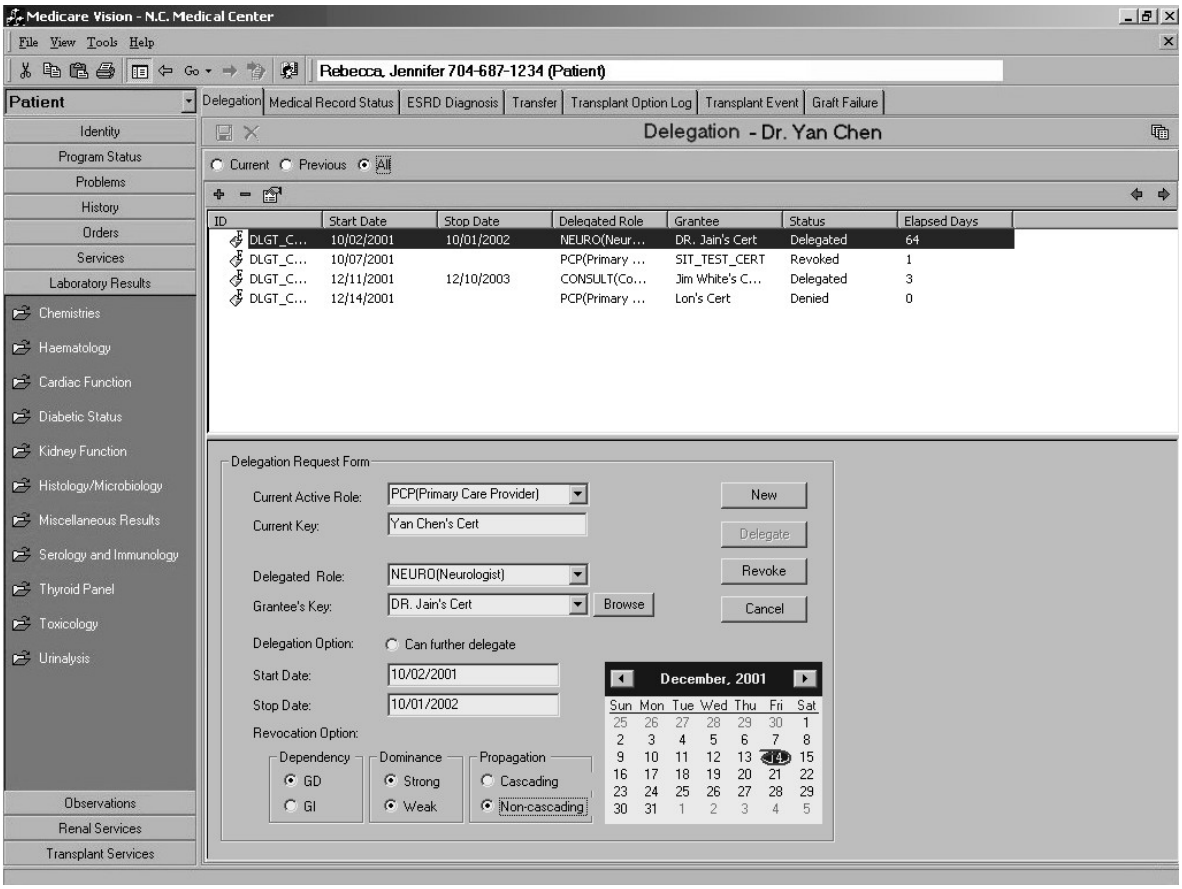

Figure 9: Delegation and revocation user interfaces in a healthcare information system

4.3.1 Delegation Within Organization

We first show an example delegation scenario in which Dr. *Chen*, as a Neurologist, is authorized to delegate his role authority NEURO to Dr. *Jain*, based on the delegation authorization rule 1 and the delegation authorization derivation rule described earlier. Dr. *Chen* initiates the delegation scenario by sending a request to the Reference Monitor to delegate the role NEURO to Dr. *Jain*. The Reference Monitor decomposes the delegation request into a role-based delegation. The delegation agent takes the information from the Reference Monitor and evaluates the delegation request through applying the delegation authorization derivation rule. In this case, it checks Dr. *Chen*'s current activating role NEURO using the active function call and enforce the delegation authorization derivation rule logic. After successful authorization, Dr. *Jain* is assigned to role NEURO.

The sequence of message exchanged and the graphic user interface for role-based delegation are shown in figure 8 and 9 respectively. We have integrated our delegation framework with the existing healthcare information system. When user clicks the delegation property page, all previous delegations and their information requested by the user are displayed. User can initiate a new delegation using the request form. Also user can revoke an existing delegation.

The delegation from Dr. *Jain* to Dr. *Chen* with the role GYNECO is a dual of the above scenario.

4.3.2 Discussion of Delegation Across Organizations

While our initial framework was designed to depict the structure of delegation in a single organization, the framework can be extended to support delegation across organizations. There are two major points concerning the cross-organization delegation: how to establish a trust between an external party and the information system? And how to define a policy to authorize the action? Managing trust between collaborative organizations using RBAC has been explored by [17]. To fully address these issues is beyond the scope of this paper. It is our future work to explore the concepts and architecture for cross-organization authorization through role-based delegation. In this section we briefly discuss our current approaches.

Currently, we use X.509 certificates to establish the trust. A user is trusted if the certificates is signed by a trusted certificate authority. The specific access privileges a user assumes in the healthcare information systems are determined by role memberships associated with his/her public key. The entitlement between roles and public key has been maintained by a role server. A rule can be defined to specify the authorization of a cross-organization delegation or revocation. But few restrictions can be imposed on such a rule at this moment. The system may not know the role memberships of a user in another organization; even if it knows, the information may not mean anything to the system. Although a mapping function can be provided for role mapping between two trusted organizations, we trust users to exercise discretion in a cross-organization delegation. In the virtual hospital example, Dr. *Chen*, as a PCP of patient *Jennifer*, is authorized to delegate the role authority CONSULT (which is junior to role PCP) to a trusted external user by his discretion, in this case, to Dr. *White* in a specialist clinic, based on the delegation authorization rule 3. The same principle can be applied to the delegation between medical researchers.

4.3.3 Revocation

Several schemes are identified for revocation of role-based delegation in [3]. The revocation scenarios might be quite complicated in certain cases. We only consider a simple revocation scenario in this paper. In the virtual hospital example, after *Jennifer* has successfully delivered her child, Dr. *Chen* should no longer share specific neurological information about *Jennifer* with Dr. *Jain*. Thus Dr. *Jain* should be revoked from role NEURO. The interactions and sequence of message exchanged in the revocation scenario is similar to the delegation scenario illustrated in figure 8. The revocation request form is same as a delegation form as shown in figure 9.

4.4 Delegation and Revocation Audit

The benefits of sharing medical information at some points will come into conflict with patient privacy. Access controls alone are not enough. Although we assume that users can be trusted to exercise discretion in how they use resources, we cannot simply neglect the possibilities of security breaches. Healthcare information systems themselves have severe audit requirements. This is for both safety and medico-legal reasons. For example, access to medical records should be logged with the user's name, as well as date and time; all delegation and revocation actions should be marked on the audit trail. A security officer can review these access records and audit trails periodically, so that breaches can be traced and detected.

5. CONCLUSION AND FUTURE WORK

Sharing of information resources is a key factor to substantial improvements in productivity and quality of services. In this paper we have implemented a role-based delegation framework to manage information sharing in the healthcare information system. The central idea is to use delegations as a means to propagate access to protected resources by trusted users. We presented the architecture and described our implementation for the delegation framework. A key feature to enhance the administrative operations of the framework is a rule editor which allows us to manage delegation and revocation rules.

In the current implementation, we focused on a healthcare setting requirements for the delegation framework. We believe our approach can be utilized to support any collaborative environments. It is our future work to extend our framework to support information sharing in other environments, such as academic research institutes, government and commercial organizations. In addition, we are now experimenting with representing rules with XML based languages and investigate signed XML statements [11] instead of X.509 certificates to bind both user identity and role attributes. We would also study how we can distribute and manage rules across organizational boundaries to improve the efficiency of administration and to fully enable delegation and revocation procedures.

Reference

[1] G. Wiederhold, M. Bilello, V. Sarathy, and X. Qian: "Protecting Collaboration"; Proceedings of the NISSC'96 National Information Systems Security Conference, pages 561-569. Oct. 1996.

[2] M. Gasser, E. McDermott. An Architecture for Practical Delegation a Distributed System. 1990 IEEE Computer Society Symposium on Research in Security and Privacy. Oakland, CA, May 7-9,1990

[3] L. Zhang, G. Ahn, and B. Chu. A Rule-Based Framework for Role-Based Delegation. Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT), pages 153–162. Chantilly, VA, May 3-4, 2001

[4] R. Sandhu, E. Coyne, H. Feinstein and C. Youman. Role-based access control model. IEEE Computer, 29(2), Feb. 1996.

[5] R. K. Thomas, R. S. Sandhu. Task-based Authorization Control (TBAC): A Family of Models for Active and Enterprise-oriented authorization Management. In proceedings of the IFIP WG11.3 Workshop on Database Security, Aug. 1997

[6] D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn. A role-based access control model and reference implementation within a corporate intranet. ACM Transactions on Information and System Security, 2(1):34-64, Feb 1999.

[7] S. I. Gavrila, J. F. Barkley. Formal specification for role based access control user/role and role/role relationship management. Proceedings of the third ACM workshop on Role-based access control, pages 81-90, Fairfax, VA, Oct. 22-23, 1998

[8] R. T. Simon and M. E. Zurko. Separation of duty in role-based environments. Proc. Foundations of Compute Security Workshop, July 1997

[9] C. K. Georgiadis, I. Mavridis, G. Panglos, and R. K. Thomas. Flexible team-based access control using contexts. Proceedings of ACM Symposium on Access Control Models and Technologies, pages 21–27. Chantilly, VA, May 3-4, 2001

[10] ITU-T Recommendation X.509. Information technology - Open systems Interconnection - The Directory: Authentication Framework, 1997.

[11] XML-Signature Syntax and Processing, W3C Candidate Recommendation 31-October-2000, http://www.w3.org/TR/xmldsig-core/

[12] G. Potamias, M. Tsiknakis, D. Katehakis, E. Karabela, V. Moustakis, and S. Orphanoudakis. Role-Based Access to Patients Clinical Data: The InterCare Approach in the Region of Crete. Proceedings of MIE 2000 and GMDS 2000, IOS Press, pages 1074-1079, Hannover, Germany, August 27- September 1, 2000.

[13] R. Anderson. A Security Policy Model for Clinical Information Systems. IEEE Symposium on Security and Privacy, pages 30-45. May 6-8, 1996

[14] J. J. Longstaff, M. A. Lockyer, and M. G. Thick. A model of accountability, confidentiality and override for healthcare and other applications. Proceedings of the 5th ACM workshop on Role-Based Access Control, pages 71-76, Berlin, Germany. July 26-27, 2000

[15] J. Poole, J. Barkley, K. Brady, A. Cincotta, and W. Salamon. Distributed Communication Methods and Role-Based Access Control for Use in Health Care Applications. http://hissa.ncsl.nist.gov/rbac/

[16] L. Giuri and P. Iglio. Role templates for content-based access control. In 2nd ACM Workshop on Role-Based Access Control, pages 153-159, Fairfax, Virginia, November 1997.

[17] T. Hildmann, J. Barholdt. Managing trust between collaborating companies using outsourced role based access control. Proceedings of the 4th ACM workshop on Role-Based Access Control, pages 105-111. October 28-29, 1999, Fairfax, Virginia, US

[18] Thomas KR. Team-Based Access Control (TMAC): A Primitive for Applying Role-Based Access Controls in Collaborative Environments. Proceedings of the 2nd ACM workshop on Role-based access control. Fairfax, VA USA; 1997