# Authorization Management for Role-based Collaboration*

**Gail-Joon Ahn, Longhua Zhang, Dongwan Shin** and **Bill Chu**
University of North Carolina at Charlotte
Charlotte, NC, U.S.A.
{gahn,lozhang,doshin,billchu}@uncc.edu

**Abstract** – *Information sharing among collaborating organizations usually occurs in broad, highly dynamic network-based environments, and formally accessing the resources in a secure manner poses a difficult challenge. The mechanisms must be provided to protect the resources from adversaries. The proposed delegation framework addresses the issue of how to advocate selective information sharing among collaborating organizations. We introduce a systematic approach to manage delegated privileges with the specification of delegation and revocation policies using a set of rules. We demonstrate the feasibility of our approach by providing a proof-of-concept implementation. We also briefly discuss several issues from our experiment including future directions.*

**Keywords**: Role-based, authorization, collaboration, delegation

## 1 Introduction

The Internet is uniquely and strategically positioned to address the needs of a growing segment of population in a very cost-effective way. It provides tremendous connectivity and immense information sharing capability which the organizations can use for their competitive advantage. Several organizations have transited from their old and disparate business models based on ink and paper to a new, consolidated ones based on digital information on the Internet. However, information sharing on the Internet usually occurs in broad, highly dynamic network-based environments, and formally accessing the resources in a secure manner poses a difficult challenge. Balancing the competing goals of collaboration and security is difficult because interaction in collaborative systems is targeted towards making people, information, and resources available to all who need it, whereas information security seeks to ensure the integrity of these elements while providing it only to those with proper authorization.

We first address some examples in the healthcare setting to clarify the problem. 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. Suppose that 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, stim-

*0-7803-7952-7/03/$17.00 © 2003 IEEE.

ulated, and promoted as strongly as possible. However, preservation of confidentiality and respect 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 among collaborating organizations.

The rest of the paper is organized as follows. In section 2, we discuss the related works. In section 3 we discuss our approach based on existing models. Implementation details are described in section 4. Section 5 discusses the lessons learned from our experiment and concludes the paper.

## 2 Related Works

Historically, the access control problem has been couched based on subjects and objects [13]. 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 [11, 13]. 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) [12] and task-based access control (TBAC) [13] 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 [11]. 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 [14]. 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 [1]. Projects have been undertaken to explore the use of RBAC and identify sample RBAC policies in healthcare information systems [8]. 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 Our Approach

In order to deal with the aforementioned issues, our work, called FRDIS (A Framework of Role-based Dele-

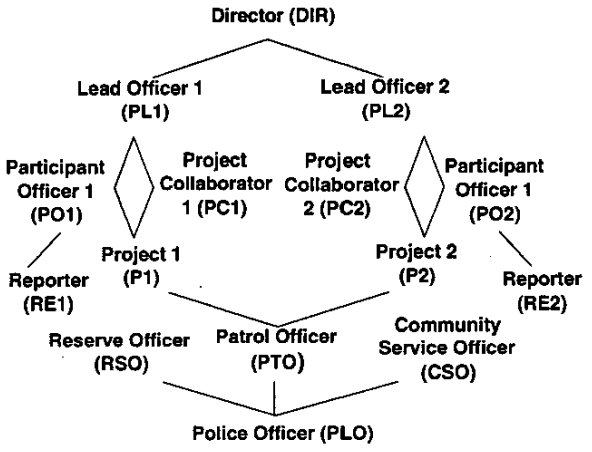


Figure 1: Role Hierarchy and Membership

Table 1: Role Membership

| ROLES | DIR | PL1 | PL2 | PO1 | PO2 |
|---|---|---|---|---|---|
| USERS | John | Deloris | Cathy | Michael<br>David | Mark<br>Lewis |

gation for Information Sharing), leverages the existing models [11, 14]. To illustrate each functional component in our model, we use the role hierarchy example illustrated in Figure 1 and Table 1.

To simplify the discussion of delegation, we assume a user cannot be delegated to a role if the user is already a member of that role. For example, project leader Deloris with role PL1 cannot be delegated the role PO1 or PC1 since he has already been an implicit member of these roles.

## 3.1 Role Delegation

We first define a new relation called delegation relation (DLGT). 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. In a user-to-user delegation, there are four components: a delegating user, a delegating role, a delegated user, and a delegated role. For example, (*Deloris*, PL1, *Cathy*, PL1) means *Deloris* acting in role PL1 delegates role PL1 to *Cathy*. A delegation relation is one-to-many relationship on user assignments. The delegation relation supports role hierarchies: a user who is authorized to delegate a role r can also delegate a role $r'$ that is junior to $r$. For example, (*Deloris*, PL1, *Lewis*, PC1) means *Deloris* acting in role PL1 delegates a junior role PC1 to *Lewis*. A delega-

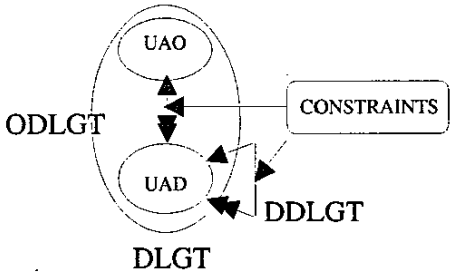


Figure 2: Delegation Relation

tion relation is one-to-many relationship on user assignments. It consists of original user delegation (ODLGT) and delegated user delegation (DDLGT). Figure 2 illustrates components and their relations in FRDIS. We assume each delegation relation may have a duration constraint associated with it. If the duration is not explicitly specified, we consider the delegation as permanent unless another user revokes it. The function Duration returns the assigned duration-restriction constraint of a delegated user assignment. If there is no assigned duration, it returns a maximum value.

FRDIS has the following components and theses components are formalized from the above discussions.

- T is a set of duration-restricted constraint.
- DLGT $\subseteq$ UA $\times$ UA is one to many delegation relation. A delegation relation can be represented by $(u, r, u', r') \in$ DLGT, which means the delegating user u with role r delegated role $r'$ to user $u'$.
- ODLGT $\subseteq$ UAO $\times$ UAD is an original user delegation relation.

- DDLGT $\subseteq$ UAD $\times$ UAD is a delegated user delegation relation.
- DLGT = ODLGT $\cup$ DDLGT.

In some cases, we may need to define whether or not each delegation can be further delegated and for how many times, or up to the maximum delegation depth. We introduce two types of delegation: 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 maximum delegation depth equal to one.

Also, we have an additional concept, delegation path (DP) that is an ordered list of user assignment relations generated through multi-step delegation. A delegation path always starts from an original user assignment. We use the following notation to represent a delegation path.

$uao_0 \rightarrow uad_1 \rightarrow uad_i \rightarrow uad_n$

Delegation paths starting with the same original user assignment can further construct a delegation tree. A delegation tree (DT) expresses the delegation paths in a hierarchical structure. Each node in the tree refers to a user assignment and each edge to a delegation relation. The layer of a user assignment in the tree is referred as the delegation depth. The function Prior maps one delegated user assignment to the delegating user assignment; function Path returns the path of a delegated user assignment; and function Depth returns the depth of the delegation path.

Constraints are an important aspect of RBAC 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 [5]. In FRDIS, 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.

## 3.2 Role Revocation

Several different semantics are possible for user revocation. Hagstrom and others [6] categorized revocations into three dimensions in the context of owner-based approach : global and local (propagation), strong and weak (dominance), and deletion or negative (resilience). Barka and Sandhu [3] further identified user grant-dependent and grant-independent revocation (grant-dependency) . Since negative authorization is not considered in FRDIS, we articulate user revocation in the following dimensions: grant-dependency, propagation, and dominance. Grant-dependency refers to the legitimacy of a user who can revoke a delegated role. Grant-dependent revocation means only the delegating user can revoke the delegated user from the delegated role membership. Grant-independent revocation means any original user of the delegating role can revoke the user from the delegated role. Dominance refers to the effect of a revocation on implicit/explicit role memberships of a user. A strong revocation of a user from a role requires that the user be removed not only from the explicit membership but also from the implicit memberships of the delegated role. A weak revocation only removes the user from the delegated role (explicit membership) and leaves other roles intact. Strong revocation is theoretically equivalent to a series of weak revocations. To perform strong revocation, the implied weak revocations are authorized based on revocation policies. However, a strong revocation may have no effect if any upward weak revocation in the role hierarchy fails [10]. Propagation refers to the extent of the revocation to other delegated users. A cascading revocation directly revokes a delegated user assignment in a delegation relation and also indirectly revokes a set of subsequent propagated user assignments. A non-cascading revocation only revokes a delegated user assignment.

Our preliminary study shows grant-dependent revocation for brevity. Suppose the revocation in Figure 3 is weak non-cascading, for *John* to revoke *Cathy* from role PL1, it is important to note that only *Cathy*'s membership of role PL1 is changed; other role memberships of *Cathy* and all the delegated user assignments propagated by *Cathy* are still valid. If the revoked node is not a leaf node, non-cascading revocation may leave a "hole" in the delegation tree. A solution might be the revoking user takes over the delegating user's responsibility. In this example, *John* takes over the delegating user's responsibility from *Cathy*, and changes all delegation relations: (*Cathy*, PL1, $u$, $r$) $\in$ DLGT to (*John*, DIR, $u$, $r$) $\in$ DLGT. In this case, *John* takes over *Cathy*'s delegating responsibility for *Mark* and *Lewis*.

## 3.3 Rule-Based Policy Specification Language

FRDIS 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 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

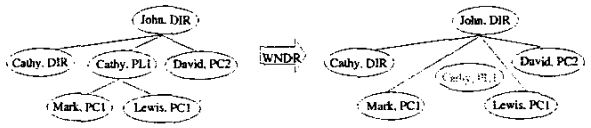


Figure 3: Weak Non-cascading Revocation

and revocation policies; authorization derivation rules enforce these policies in the healthcare information system; and integrity rules specify and enforce role-based constraints.

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

$can\_delegate(r, cr, n) \leftarrow .$

*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:

$$\begin{aligned} &der\_can\_delegate(u, r, u', r', dlg\_opt) \leftarrow \\ &\quad can\_delegate(r'', cr, n)\& \\ &\quad active(u, r, s)\& \\ &\quad delegatable(u, r)\& \\ &\quad senior(r, r'')\& \\ &\quad in(u', cr)\& \\ &\quad junior(r', r'')\& \\ &\quad in(depth(u, r), n). \end{aligned}$$

*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 are also defined for role-based revocations and are applied to specify constraints.

# 4 Implementation Details

The notions described in FRDIS and the rule-based policy specification language are designed to be utilized within an administrative-directed delegation management architecture. An overview of the proposed architecture is shown in Figure 4. 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. Since delegation and revocation services are only 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.

The role service is provided by a role server, which is an implementation of the RBAC96 and FRDIS components. A role server maintains RBAC database and provides user credentials, role memberships, associated permissions, and delegation relations of the system. 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. They are implemented as authorization policies that authorize requests from users. 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. 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. The reference monitor makes access control decisions based on information supplied by the access control agent. In large role-based system, there may be tens or hundreds of delegation and revocation rules. The rule editor is developed to simplify the management of these rules. As a portion of an integrated RBAC administration platform developed to manage various RBAC and FRDIS components, the rule editor is used to view, create, edit, and delete delegation and revocation rules.

Our implementation leverages FRDIS features and X.509 attribute certificate. We attempt to implement the proof-of-concept prototype implementation of FRDIS on privilege management infrastructure (PMI) [7]. PMI provides certificate-based authorization with attribute certificates while public-key infrastructure (PKI) does certificate-based authentication with public-key certificates, so called identity certificates. One of the great benefits of PMI is to establish the trustiness among different authorization domains as long as each domain keeps the meaning of attributes intact. Thus, access control could be enforced not just within a single authorization domain, but also across multiple domains [2, 9].

Three components are identified for managing attribute certificates: privilege asserter, privilege verifier, and PMI attribute authority. Two different attribute

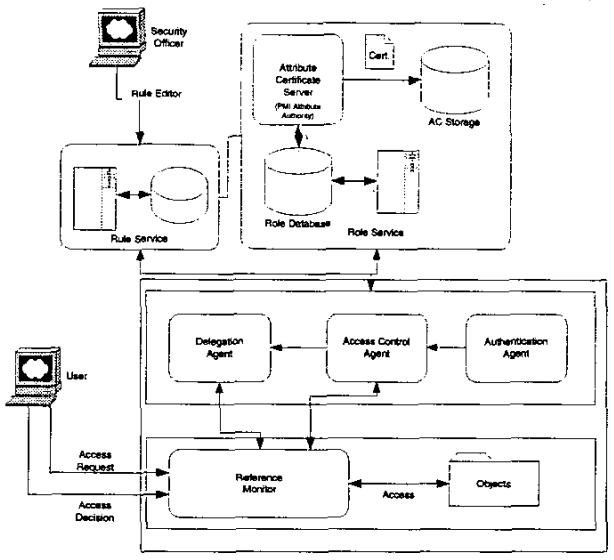


Figure 4: FRDIS Architecture

certificates are employed: role assignment attribute certificate (RAAC) for assigning roles to a user and role specification attribute certificate (RSAC) to assign specific permissions to a role. Our tasks are divided into two phases. The first phase is to build APIs for both a role-based decision making engine and attribute certificates. Those APIs are the core building blocks for constructing an access control policy server and an attribute certificate server. The second phase is to implement each entity integrating with APIs. Currently we are in the transition period from the first phase to the second.

**Privilege asserter** is a client. The client is a user or a system. It asks for and retrieves RAACs from PMI attribute authority and requests access to web services (or protected resources). We developed a simplified privilege asserter using ActiveX control, named attribute certificate manager. The manager enables a user to import downloaded BER-encoded RAACs into Windows registry. It also allows the user to view and select on of RAACs in the registry. The selected RAAC will be presented for requesting access to resources. We use Microsoft Internet Explorer (Version 6.0) to activate the ActiveX control-based privilege asserter.

**Privilege verifier** is composed of server, access control policy server, and policy database. The server is a protected resource server or an application server. When a client wants to access the server, the server asks the access control policy server if the client has the privilege to access what it requests. The access control policy server makes access control decisions based on both the client's roles from a RAAC and the permissions assigned to the roles from a RSAC. The RSAC can be obtained from the PMI attribute authority or the policy database. The policy database maintains all RSACs that are previously retrieved from the PMI attribute authority. Internet Information Server (Version 5.0) is used as a server. An HTTP raw data filter, called AC filter, was developed using Microsoft ISAPI (Internet Server API) technology. Its main task is screening the incoming raw data from a client to see if the client presents any attribute certificate.

We also developed an application working as an access control policy server. This application has been developed in C++. An engine for making access control decisions is a major component in this application. After receiving a valid RAAC and requested objects (with operation type) from the web server, the engine extracts permissions from the RSAC and checks if the requested object (with operation type) is in the list of permissions. The programming library, called RBAC API, was developed to facilitate such procedures.

**PMI attribute authority** has four entities: attribute certificate server, AC storage, role database, and role engineering administration. The attribute certificate server signs and issues both RAACs and RSACs. After issuing those certificates, it stores them into a publicly accessible repository, AC storage. Private role database retains all components required to construct a role-based infrastructure and is used for role engineering, which is referred to as an approach to defining roles and assigning permissions to the roles [4]. A simple version of attribute certificate server was developed in C++ to generate RAACs and RSACs. The programming library, called AC SDK, was built for supporting the functionality related to the generation of the at-

tribute certificates. Netscape Directory Service 5.0 was used for both a role database and an AC storage.

# 5 Conclusion

In this paper we have implemented a role-based delegation framework to manage information sharing for collaborating organizations. 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 the rule specification which allows us to manage delegation and revocation policies. 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 critical infrastructures.

## Acknowledgment

This work was partially supported at the Laboratory of Information of Integration, Security and Privacy at the University of North Carolina at Charlotte by the grants from National Science Foundation (NSF-IIS-0242393).

# References

[1] R. Anderson. A security policy model for clinical information systems. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 30–45, Oakland, CA, May 6-8 1996. IEEE.

[2] Gail-Joon Ahn, Ravi Sandhu, Myong Kang, and Joon Park. Injecting RBAC to secure a web-based workflow system. In *Proceedings of 5th ACM Workshop on Role-Based Access Control*, Berlin, Germany, July 26-27 2000. ACM.

[3] E. Barka and R. Sandhu. Framework for role-based delegation model. In *Proceedings of 23rd National Information Systems Security Conference*, pages 101–114, Baltimore, MD, October 16-19 2000.

[4] E. Coyne. Role engineering. In *Proceedings of 1st ACM Workshop on Role-Based Access Control*, Gaithersburg, MD, November 1995. ACM.

[5] David F. Ferraiolo, John F. Barkley, and D. Richard Kuhn. A role based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security*, 2(1), February 1999.

[6] A. Hagstrom, S. Jajodia, F. P. Presicce, and D. Wijesekera. Revocations - a classification. In *Proc. 14th IEEE Computer Security Foundations Workshop*, pages 44–58, Nova Scotia, Canada, June 2001.

[7] ITU. *ITU-T Recommendation X.509. Information Technology: Open Systems Interconnection - The Directory: Public-Key And Attribute Certificate Frameworks*, 2000. ISO/IEC 9594-8.

[8] G. Potamias, M. Tsiknakis, D. Katehakis, E. karabela, V. Moustakis, and S. Orphanoudakis. Role-based access to patient clinical data: InterCare approach in the region of Crete. In *Proceedings of MIE and CMDS 2000*, pages 1074–1079, Hannover, Germany, August 27 - September 1 2000. IOS Press.

[9] Dongwan Shin, Gail-Joon Ahn, and Sangrae Cho. Role-based EAM using x.509 attribute certificate. In *Proceedings of Sixteenth Annual IFIP WG 11.3 Working Conference on Data and Application Security*, Cambridge, UK, July 29-31 2002. IFIP.

[10] Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security*, 2(1), February 1999.

[11] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[12] Roshan Thomas. Team-based access control (tmac). In *Proceedings of 2nd ACM Workshop on Role-Based Access Control*, pages 13–19. ACM, Fairfax, VA, November 6-7 1997.

[13] Roshan Thomas and Ravi Sandhu. Task-based authorization controls (TBAC): Models for active and enterprise-oriented authorization management. In T. Y. Lin and Xiaolei Qian, editors, *Database Security XI: Status and Prospects*. North-Holland, 1997.

[14] Longhua Zhang, Gail-Joon Ahn, and Bill Chu. A rule-based framework for role-based delegation. In *Proceedings of 6th ACM Symposium on Access Control Models and Technologies*, pages 153–162, Chantilly, VA, May 3-4 2001. ACM.