Role-Based Security for Distributed Object Systems

Nicholas Yialelis Emil Lupu Morris Sloman

Department of Computing, Imperial College
180 Queen's Gate, London SW7 2BZ, United Kingdom
Email: {n.yialelis, e.c.lupu, m.sloman}@doc.ic.ac.uk

Abstract

This paper describes a security architecture designed to support role-based access control for distributed object systems in a large-scale, multi-organisational enterprise in which domains are used to group objects for specifying security policies. We use the concept of a role to define access control related to a position within an organisation although our role framework caters for the specification of both authorisation and obligation policies. Access control and authentication is implemented using security agents on a per host basis to achieve a high degree of transparency to the application level. Cascaded delegation of access rights is also supported. The domain based authentication service uses symmetric cryptography and is implemented by replicated servers which maintain minimal state.

1 Introduction

Distributed systems are increasingly being used in commercial environments necessitating the development of trustworthy and reliable security mechanisms. These must prevent unauthorised access to resources, compromise of information integrity or loss of confidentiality. Distribution requires the need for encryption over untrusted networks and remote computers may not be fully trusted. Distributed systems may contain millions of objects and cross inter-organisational boundaries forcing decentralisation of security management. There is often no clear informal or formal specification of enterprise authorisation policies and no tools to translate policy specifications to access control implementation mechanisms such as capabilities or Access Control Lists. It is thus difficult to analyse the policy to detect conflicts or flaws and it is difficult to verify that the implementation corresponds to the policy specification.

This paper gives the overview of a security architecture to support an access control model that makes use of domains, explicit policy objects and roles. *Domains* [1] are used to group objects to which a common policy applies, to partition management responsibility in large systems or for the convenience of humans (c.f. file system directories). Domains can contain subdomains to form a hierarchical structure. A *role* identifies the rights, duties, functions and interactions, associated with a position such as vice president, board director, security administrator, doctor or nurse in a hospital. We model *rights* as *authorisation policies* which specify what activities a subject is permitted (or forbidden) to perform on a set of target objects. *Duties* are modelled as *obligation policies* which specify what activities a subject must or must not perform on a set of target objects. A role is the set of authorisation and obligation policies which have a particular role position as a subject [2], although we focus only on authorisation policies in this paper. The advantage of using roles for specifying enterprise policies is that individuals can be assigned to or withdrawn from the role positions without having to respecify the policies applying to the role. An object oriented approach to specifying roles permits multiple instances of a basic role to be instantiated e.g. for each nurse in a hospital.

We provide a framework in which roles and inter-role relationships represent enterprise security requirements. Roles are expressed in terms of policies which refer to domains to provide a very flexible basis for specifying Role Based Access Control [3] which caters for large scale, inter-organisational distributed systems. High-level abstract policies can be defined and then refined into a number of implementable policies. Further information on roles can be found in [4] while this paper concentrates on the security aspects of the architecture

The security architecture enforcing the access control consists of four main building blocks: the domain service, the policy service, the authentication service and the security agents which are employed on each host to achieve a high degree of transparency of the security services to the application level. The distributed domain service (discussed in section 2.1) maintains information on domain membership and the policies applying to domains. The policy service (see section 3.1) allows specification and refinement of policies by human users. These policies are then disseminated to the distributed agents which implement them. The access control model is described in Section 2 as well as more detail of how roles are specified in terms of domains and policies. Section 3 describes the security agents needed in each host for authentication and access control.

1080-1383/96 $5.00 © 1996 IEEE
Proceedings of WET ICE '96

2 Access control policies and roles

Our access control model is based on subjects invoking operations on targets where the subject or target may be active or passive objects. Objects may be representatives of a user (c.f. login shell), distributed agents acting on behalf of users, devices, files or components of a service. An object has a unique Object Identifier (OID) which contains the network address of the object and an element which is always unique.

2.1 Domains

In a large system many objects or users may exhibit common characteristics with respect to some criteria, so it is useful to specify policies that apply to a *group* of objects rather than individual ones. A domain is an object which maintains a list of references to objects that have been explicitly grouped together to reflect the needs of users or management [2]. Domains provide similar functionality to *groups* in traditional access control systems but are used to group targets as well subjects. If a domain holds a reference to an object, the object is said to be a direct member of that domain and the domain is said to be its parent. Domains can also be nested in that a subdomain may be a member of a parent domain and the subdomain members are then indirect members of the parent. An object can be a direct or indirect member of multiple domains.

Multiple domain servers store the domain membership information and a domain server is trusted to certify membership of objects in the domains it maintains. The domain servers may be managed by principals that represent different interests and so are trusted to different extents. The access control decision making, which is based on the domain membership of the objects, cannot rely on a single authority certifying this membership.

2.2 Access control policies

We consider a policy in its simplest form to be a relationship between a subject and a target. An *obligation policy* determines what operations the subject must (or must not) invoke on the target object, but are beyond the scope of this paper – see [5] for further details.

An *authorisation or access control policy* determines what operations the subject is permitted (positive authorisation) or forbidden (negative authorisation) to perform on the target. We allow the use of negative authorisation policies at the specification level as they are found in enterprise policies e.g. students are not permitted to reboot the workstations. The security architecture only enforces positive policies with the assumptions that actions not explicitly authorised are forbidden and that a set of policies containing negative authorisation policies can be refined into a set containing only positive ones.

A policy specifying abstract activities can be refined into a hierarchy of more specific abstract policies and eventually into enactable leaf-level policies that can be enforced by the access control agents. An implementable positive access control policy specifies the operations that objects in the subject domain are permitted to perform on objects in the target domain e.g. operations op1, op2 on objects of type T1, and operations op1, op3 on object of type T2. In addition, a policy may impose constraints such as a validity time for the policy (e.g. 09–17.00) or required encryption for the invocation and reply data [1].

By default, policies propagate to subdomains and hence to indirect members of parent domains. In some cases it is useful to be able to restrict the propagation to only direct members or to certain combinations of subdomains. We use *domain scope expressions* to specify groups of objects in terms of set operations on domains and objects e.g. *DomA ^ *DomB specifies the objects that are direct or indirect members of both DomA *and* DomB.

The notion of the access control policy has been extended to deal with the *delegation of access rights* whereby an object authorises a possibly remote agent to act on its behalf to access a service. Objects are not all equally trusted so it is necessary to control what access rights can be delegated to which objects and the possession of a right does not imply the right to delegate it. For these reasons, there is a need for a policy scheme that enables the determination of: i) *The Grantor*, i.e. who can delegate, ii) *The Grantee*, i.e. to whom rights can be delegated, iii) *The Delegatable rights*, i.e. what rights the grantor is permitted to delegate to the grantee.

An *Extended Access Control Policy*, (Fig. 2.1) is used to control delegation of access rights. It determines that objects in the Subject scope can delegate to objects in the Grantee scope the right to perform operations, specified in the Operations field, on objects in the Target scope.

Cascaded delegation is allowed provided that all grantees are in the Grantee scope of the policy. Our scheme does not impose constraints on the order of the delegation steps (c.f. [6]).

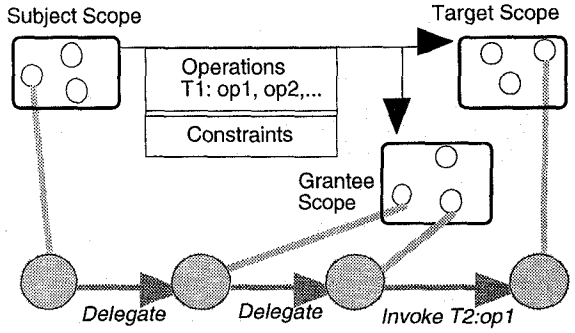

Figure 2.1: Graphical representation of an Extended Access Control Policy

2.4 Roles

We use the concept of a Role associated with a position so that policies can be specified with respect to organisational positions and describe the duties and access

rights of the individuals assigned to the positions. This means that the policies do not have to be respecified when individuals are assigned to new positions.

A role is composed of a *Role Position Domain* (RPD) and a *set of access control and obligation policies* which have the RPD as a subject. An RPD represents a position or status within the organisation, for instance security administrator, secretary, doctor [7]. The organisational structure, however, cannot be fully represented by roles since individuals interact and co-operate with each other. The responsibility of the individuals assigned to roles includes the relationships to other roles e.g. supervision of work. These relationships include the policies between related roles (e.g. Ward10_doctor is permitted to assign tasks to Ward10_nurse) and when and how they can access shared resources such as patient files. Relationships also include the interaction protocols defining the exchange of information and the concurrency constraints on the activities of the related roles. Thus, our framework identifies for a role position: i) the access control and obligation policies related to target objects, ii) the interactions between roles which reflect organisational role relationships, and iii) both intra- and inter-role concurrency constraints. The complete role model is defined in more detail in [4].

This paper only deals with simplified security roles composed of a RPD and a set of positive access control policies. A generic role can be specified as a role class in terms of policies which define a set of permitted activities e.g. for a nurse. These could be further specialised e.g. for a surgical nurse [8]. Multiple instances of surgical nurses could be created each with their own position and target domains. The nurse assigned to Ward10 would be responsible for a different set of target patients from the nurse assigned to Ward9. The access control policies associated with a role i.e. which have the RPD as subject domain define the access privileges of the individuals assigned to that role. Individuals can be assigned to or removed from a role without changing the policies and relationships specified for the role.

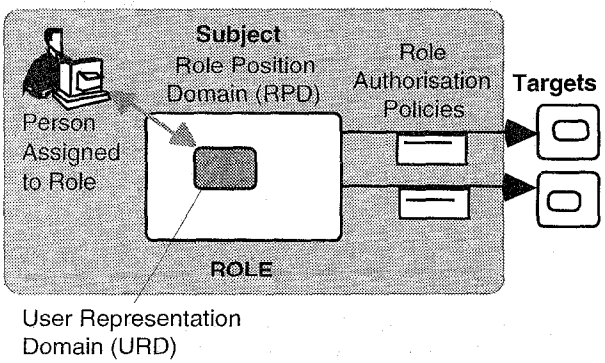

Figure 2.2: A human manager assigned to a role

A user is permanently represented within the system by a *User Representation Domain* (URD). The user is assigned to a position and thus to the corresponding role by including her URD in the RPD (see Fig. 2.2). Thus, policies associated with the role will propagate and apply to the URD representing the user. Note that there may be propagated policies applying to a position domain (and to the URDs included in that domain) which are not a component of the role associated with that position. For example the College policy for the use of computers propagates to all the research students but is not a specific part of the authorisations of their research role. Similarly, a Professor having her URD included in other organisational domains (research groups, experts on a particular subject) will be subject to other policies than those specified within the role(s) she is assigned to. Multiple URDs may be included in a RPD to represent the sharing of a position by a number of people and a URD can be included in several RPDs if the person performs multiple roles [2, 4].

2.5 Simultaneous role sessions

When a user logs into a system, a process acting as an adapter object between the user and the system e.g. login shell is created within the URD. Note that according to Fig. 2.2 this object inherits the access rights associated with *all* the roles in which the URD is included. This implies that an action could be performed in a role with access rights inherited from another role. This is not desirable and contradicts the concept of a role as a consistent set of access rights needed to perform the role's tasks. The activities related to different roles should be presented separately to the user e.g. each role has its own window and the security system must be able to distinguish in which role the user is acting so as to only use the access rights relating to that role.

This can be achieved by keeping the URD out of the RPD and instead specifying a policy permitting the URD to create an agent within an RPD to which it has been assigned. This policy has the RPD as target and is not part of the role's policies. The adapter object in the URD is similar to an X server maintaining windows for each of the roles in which the user chooses to work. The connection shown between the adapter and agents in Fig. 2.3 does not imply remote communication – the objects could be implemented as threads within a single process but conceptually are members of different domains and so execute with the rights specific to the RPD. The menu within a window can be specific to a role and only display those operations permitted by the policies for that role. The role policies will also specify what target domains can be accessed from the role.

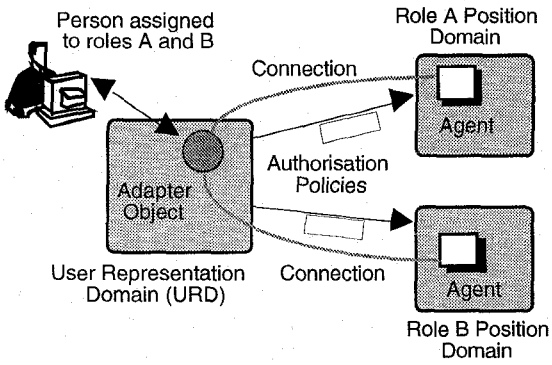

Figure: 2.3 Multiple role sessions

The role framework relies on the policy service and the domain structure. Roles are only visible at the specification level and are implemented as domains and policies for access control purposes. The security architecture does not have to know about roles and only sees domains and policies.

3 Overview of the security architecture

We take an approach to building the system in which distributed security is provided at the levels of the host manager and the application server. A Host Manager provides an *Authentication Agent* (AA) and an *Access Control Agent* (ACA) that are trusted to act on behalf of all application objects on its host and is accessed via a simple application programming interface (API). By transferring part of the security functionality into the host manager we minimise the size of the security components that must be replicated among the application servers and the number of principals (objects in this context) that have to be registered with the authentication service.

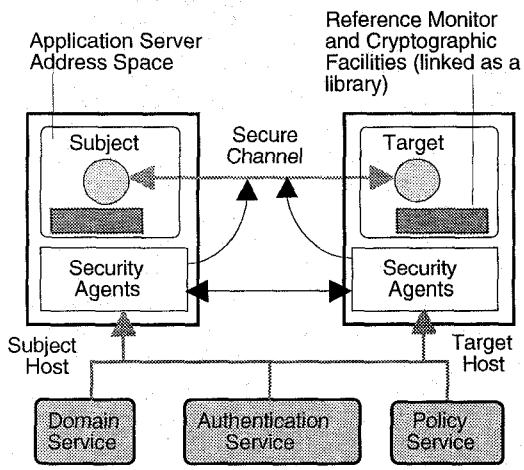

Figure 3.1: Security Architecture Overview

The security agents are responsible for authenticating remote objects, determining access control policies applying to subject-target pairs and establishing session keys to be used by subject-target pairs. The operation of these agents is supported by the Policy, the Domain and the Authentication Services (see Fig. 3.1).

3.1 Policy Service

Policy Objects are maintained within the servers of the policy service and are registered within domains. An administrator who has the necessary access privileges to the domain can create, edit, activate, disable and delete policies using a Policy Editor [5]. When an (extended) access control policy is activated, it is distributed to the ACAs of the hosts maintaining objects to which the policy applies (see Fig. 3.2). The domain service is queried to determine which objects are in the subject, target and grantee scope of the policy. The scopes of an activated policy are re-evaluated every time the domain structure changes or the membership of a non-domain object changes. In addition, when a policy is disabled, special revocation tokens are propagated down the domain hierarchy and are eventually given to the ACAs of the objects in the scopes of the disabled policy. The effect is that the ACAs are continuously aware of the policies applying to the objects being maintained on their hosts. For a more detailed description of the policy propagation mechanism see also [9].

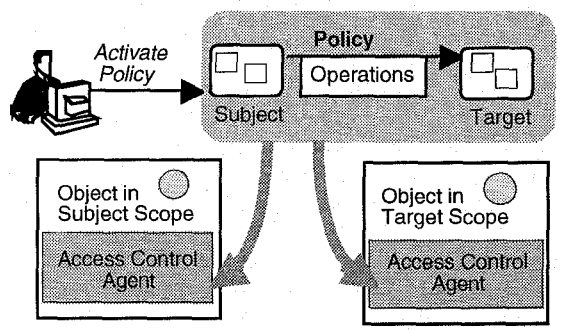

Figure 3.2: Policy Distribution to Subject and Target Access Control Agents

3.2. Authentication Service

The Authentication Service (AS) is used to authenticate users and remote authentication agents and verify the authenticity of delegation tokens and domain membership statements which form the basis for access control decisions. User authentication can be achieved either by passwords or smart cards that are capable of storing a key and performing cryptographic operations. When the identity of the user has been verified, the system makes the adapter object on the login workstation a member of the URD of the user.

The AS is transparent to the application objects as only the authentication agents (AAs) interact with it. AAs are the only objects registered with the AS. This significantly reduces the size and update rate of the AS security database. Note that a high update rate of the security database

seriously affects the performance of the system as it involves expensive cryptographic operations (c.f [10]). It is the responsibility of the user (or the object acting on her behalf) to create an application object on a node whose AA, ACA and system software/hardware can be sufficiently trusted. An AA believes that the AA and ACA on a remote host are sufficiently trustworthy to act on behalf of the application objects on that node. The secrecy and integrity of communication between servers on the same node (for example between application and host manager server) is provided by the underlying system software and hardware which must be trusted anyway.

An intra-realm authentication service that is based on symmetric cryptography and employs replicated authentication servers with minimal state is being developed. It uses Private-key Certificates [11] to disperse the security database so that the on-line authentication servers need to maintain no state apart from their master key, thus simplifying server replication as there is no state consistency problem. Further, it permits very simple, replicated tamper-proof machines to be used as authentication servers and as translators (relays) [6]. The relay function of the AS is of great importance in the security architecture as it allows domain membership verification without establishing shared secret keys between the verifier and the domain servers. Specifically, membership statements encrypted with the secret keys of the AAs of the domain servers, certifying the claimed membership, can be re-encrypted by the AS with the secret key of the verifier. In addition, the push method [6] can be employed whereby the claimant can collect the necessary certificates which can later be re-encrypted by the AS with the secret key of the verifier. A similar mechanism is used to verify the delegation tokens that are encrypted under the secret key of the grantor AAs.

Each AA shares a secret key with the AS and so authentication between AAs can be achieved by using a protocol described in [12] which is similar to that employed by the Kerberos system [13]. The main difference is that the authentication server in our system is provided with the private-key certificates that contain the secret keys of the AAs involved in the protocol.

The authentication service could be based on asymmetric cryptography which eliminates the need for network interaction with on-line authentication servers. However, it dramatically increases the encryption/ decryption time (symmetric encryption is considered to be 1000-5000 times faster than asymmetric cryptography [6]). As our system supports multiple domain membership authentication as well as delegation, a relatively large number of membership certificates and delegation tokens are generated and verified for each secure channel (see Section 3.3). Thus, use of an asymmetric cryptosystem would significantly increase the processing required for secure channel establishment.

3.3 Security Agents and secure channels

We use a *secure channel* between a subject and a target to represent authentication, cryptographic, access control and domain membership information. Each channel has a unique identifier (CHID) which can be used as a reference to the information related to the channel. The subject initiating an invocation on a target requests the establishment of a secure channel which involves both the AA and ACA. The subject and target can then exchange messages which are encrypted and decrypted by the Cryptographic facilities (CFs) in their address space. The access control decision for each invocation is made by the Reference Monitor (RM) in the address space of the target, based on channel related access control information provided by the target ACA. The cryptographic and RM facilities are linked as a library into each application server.

The type and degree of the communication security required e.g. integrity or secrecy is application dependent. Use of encryption may be precluded for performance reasons or by legislation, so the subject requesting a channel can specify the level of security that the channel should support.

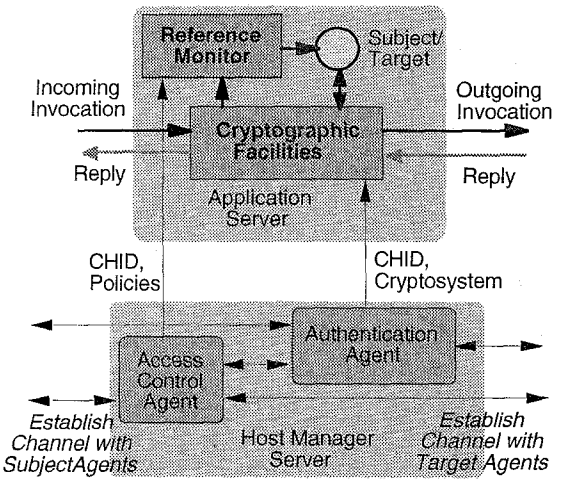

Figure 3.3: Application Server Facilities and Secure Communication

AAs establish channel keys on behalf of the application objects and can easily verify whether a remote authenticated AA is trusted to act on behalf of an object whose OID and location are known. A remote AA is always trusted to act on behalf of all objects on its host, so an AA simply checks whether the network address of the authenticated remote agent matches the address of the object with which a channel has to be established. If mutual trust between the two agents has been established, they proceed to choose a channel key and a CHID. These are given to the cryptographic facilities of the two application objects (see Fig. 3.3). The CHID generated by the subject AA is based on its host name and a sequence

number. AAs also perform verification of object membership in domains using the Domain Service. A description of the membership verification mechanism is given in [12].

The ACAs hold copies of the access control policies applying to the objects on their nodes. The target ACA determines the policies that apply to an established channel once the subject has been authenticated and its domain membership verified by the target AA. These policies are given to the RM in the address space of the target to make the final access control decision for each invocation on the established channel (see Fig. 3.3). The subject ACA determines the set of policies relating to the channel i.e. the Pseudo-Capability List which contains the OIDs of the policies applying to the subject. These are provided as a *hint* to reduce the number of policies applying to the target that have to be checked by the target ACA. A detailed description of the access control mechanism is given [9] which also describes how the mechanism has been extended to make access control decisions when delegation is involved.

4 Conclusion

We have given an overview of an access control model for distributed object systems and a security architecture that is being developed in the CORBA distributed programming environment. The access control model is based on the notion of domains to specify access control policies for groups of objects which may be subjects or targets. Our domains are more flexible and powerful than the group concept traditionally used in access control and can be used to partition responsibility or reflect the structure of large-scale enterprises. The power of the domain concept is indicated in the fact that it can be used to model positions and roles, such that the security service only has to know about domains.

Our authorisation polices provide a very flexible means of specifying access control permissions and include constraints to limit their applicability. They can also be extended to specify delegation policy. Policies explicitly identify both subjects and targets, and domains maintain information about the policies applying to them so it is easy to analyse the policies to determine those applying to a specific object.

Roles and inter-role relationships are used for the representation of the organisational structure. The role framework is combined with the ability to refine organisational policies from an abstract level to enactable rules for a full representation of the organisational structure, organisational policies and the assignment of responsibilities to managers.

The security system consists of four main building blocks. The Policy Service which maintains and distributes policy objects, the Domain Service which maintains domain objects and certifies domain membership, the Authentication Service and security agents that are employed on a per host basis. The Authentication Service is based on symmetric cryptography to minimise the encryption overhead and it is provided by on-line servers with minimal state to facilitate replication. The distributed security provided by the system is transparent to the application level and APIs are provided to facilitate the development of security unaware applications.

Acknowledgements

We gratefully acknowledge financial support from the Swiss Bank Corporation (London), EPSRC Roleman project (GR/K 37512) and BT MMN project.

References

[1] J. Moffett, M. Sloman, "User and Mechanism Views of Distributed System Management", *IEE/IOP/BCS Distributed Systems Engineering*, vol. 1, no. 1, pp. 37-47, Aug. 1993.

[2] M. S. Sloman, "Policy Driven Management for Distributed Systems," *Journal of Network and Systems Management*, vol. 2, no. 4, pp. 333-360, Dec. 1994.

[3] E. Lupu, D. Marriott, M. Sloman, and N. Yialelis, "A Policy Based Role Framework for Access Control," *First ACM/NIST Role Based Access Control Workshop*, Gaithersburg, USA, Dec. 1995.

[4] E. Lupu and M. Sloman, "Towards a Role Based Framework for Distributed Systems Management," *To appear in: Plenum Press Journal of Network and Systems Management*, vol. 5, no. 1, 1997.

[5] D. Marriott and M. Sloman, "Management Policy Service for Distributed Systems," *IEEE Third Int. Workshop on Services in Distributed and Networked Environments (SDNE'96)*, June 1996, Macau, pp. 2-9

[6] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in Distributed Systems: Theory and Practice," *ACM Trans. on Computer Systems*, vol. 10, no. 4, pp. 265-310, Nov. 1992.

[7] B. J. Biddle and E. J. Thomas, "Role Theory: Concepts and Research," New York: Robert E. Krieger Publishing Company, 1979.

[8] R. Sandhu, E. Coyne, H. Feinstein, C. Youman "Role Based Access Control Models", *IEEE Computer* vol. 29, no. 2, pp. 38-47, Feb. 1996.

[9] N. Yialelis and M. Sloman, "A Security Framework Supporting Domain-Based Access Control in Distributed Systems," *IEEE ISOC Symposium on Network and Distributed Systems Security'96*, San Diego, pp. 26-34, Feb. 1996.

[10] R. Deng, S. Bhonsle, W. Wang and A. Lazar, "Integrating Security in CORBA Based Object Architecture", *IEEE Symposium on Security and Privacy*, pp. 50-61, 1995.

[11] D. Davis and R. Swick, "Network Security via Private-Key Certificates," *ACM SIGOPS Operating Systems Review*, vol. 24, no. 4, pp. 64-67, Oct. 1990.

[12] N. Yialelis and M. Sloman, "An Authentication Service Supporting Domain Based Access Control Policies," Imperial College, Research Report DoC 95/13, Sep. 1995.

[13] C. Neuman and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 33-38, Sept. 1994.