An Authentication Service Supporting Domain Based Access Control Policies

Imperial College Research Report No. DoC 95/13

15 September 1995

Nicholas Yialelis and Morris Sloman

E-mail: ny@doc.ic.ac.uk, mss@doc.ic.ac.uk
Fax: ++44 171 581 8024

Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
UK

Abstract

This paper describes the basic architecture of an authentication service for distributed systems in which domains are used to group objects in order to specify policy. This is necessary for very large scale systems where it is impractical to specify policies for individual objects. The enforcement of a policy that is specified in terms of domains requires authentication of object membership of domains.

As the use of asymmetric cryptography would result in unacceptable performance, the proposed system is based on the use of symmetric cryptography for intra-realm authentication of identities or domain membership, while asymmetric cryptography can still be used for inter-realm authentication. It utilises replicated trusted authentication servers with minimal state in order to avoid problems in terms of the security and state consistency of the replicas. This is achieved by using private-key certificates which provide a similar functionality to the public key certificates in asymmetric cryptosystems, but have better performance. Authentication servers are also used as translators, i.e. they translate messages that were encrypted with the secret key of the sender by re-encrypting them with the secret key of the receiver. The paper also describes the establishment of secure channels between remote objects as well as the authentication of object membership of domains.

Keywords: access control, authentication, certificates, domains, security architecture, security policy.

1. INTRODUCTION

The notion of a domain is used to specify policies in terms of groups of objects rather than for individual objects. Domains provide the means of coping with the complexity of very large scale, inter-organisational distributed systems which could include millions of objects. These concepts have been used within the framework of the Esprit funded SysMan project for management of object based distributed systems [Sloman *et al.* 1993]. Access control policies specify what operations a set of subjects is permitted to perform on a set of targets. As the scope of these policies is specified in terms of domains, access control decisions are based on authenticated domain membership rather than individual object identities [Sloman 1994].

This section gives an overview of domains and how they are used to specify access control policies. It also briefly discusses the access control enforcement model and its relationship to the authentication system.

1.1 Domains

A *domain* represents a collection of objects (users, files, servers, workstations, etc.) that have been explicitly grouped together for management purposes [Sloman *et al.* 1993]. The *policy set* of a domain Dx is the set of references held by Dx to its *direct members*. An object may be direct member of more than one domain. As domains are themselves objects, they can be members of other domains. An object is an *indirect member* of a domain Dx if it is a direct member of a domain Dy which is direct or indirect member of Dx. Domain objects are maintained within *Domain Servers* which collectively provide a *Domain Service*. A Domain server holds the domain objects corresponding to a part of the domain structure and the addresses of the domain servers responsible for other parts of the domain hierarchy.

Each object has a *Unique Identifier* (UID) created from the host IP address where the object was originally created plus time stamp. This identifier remains unchanged even if the object migrates to a new host. In addition, each object has an *Object Identifier* (OID) which consists of its UID and its current network address.

1.2 Access Control Policies and Scope Expressions

An *access control policy* specifies a relationship between a *subject scope* and a *target scope* in terms of the *operations* which subjects are permitted to perform on targets as well as *constraints* on the applicability of the policy (e.g. validity time for the policy). A policy is represented by a tuple of the form:

(Subject Scope, Target Scope, Operation Set, Constraints)

The subject and target scopes in a policy are specified using *scope expressions*. They are applied on objects (of both domain or non-domain type) and return a *set of objects*. A simplified syntax of the scope expressions is defined as follows:-

```
SC_EXPR ::= object* |
            object@ |
            { object } |
            SC_EXPR + SC_EXPR |
            SC_EXPR - SC_EXPR |
            SC_EXPR ^ SC_EXPR |
            ( SC_EXPR )
```

Operators:

+ set union

- set difference[1]
^ set intersection
* when applied on a *domain object* a set is returned that contains all *direct and indirect* members of the domain and the domain object itself; otherwise a set is returned that contains the object itself.
@ when applied on a *domain object* a set is returned that contains all *direct* members of that domain; otherwise Ø is returned.
{ } returns a set that contains the object on which it is applied.

The interpretation of the expressions is from left to right.

1.3 Access Control Policy Enforcement Model

We briefly describe the access control model (shown in Figure 1.1) to indicate its relationship to the authentication service. The *Policy Service* permits human security managers to specify access control policies graphically and distributes these policies to *Access Control Agents (ACA)* which reside in every node within the system. The ACA holds the policies applying to targets within a node. A *Subject* is the object that invokes an operation and a *Target* is the object that provides an interface on which operations can be invoked.

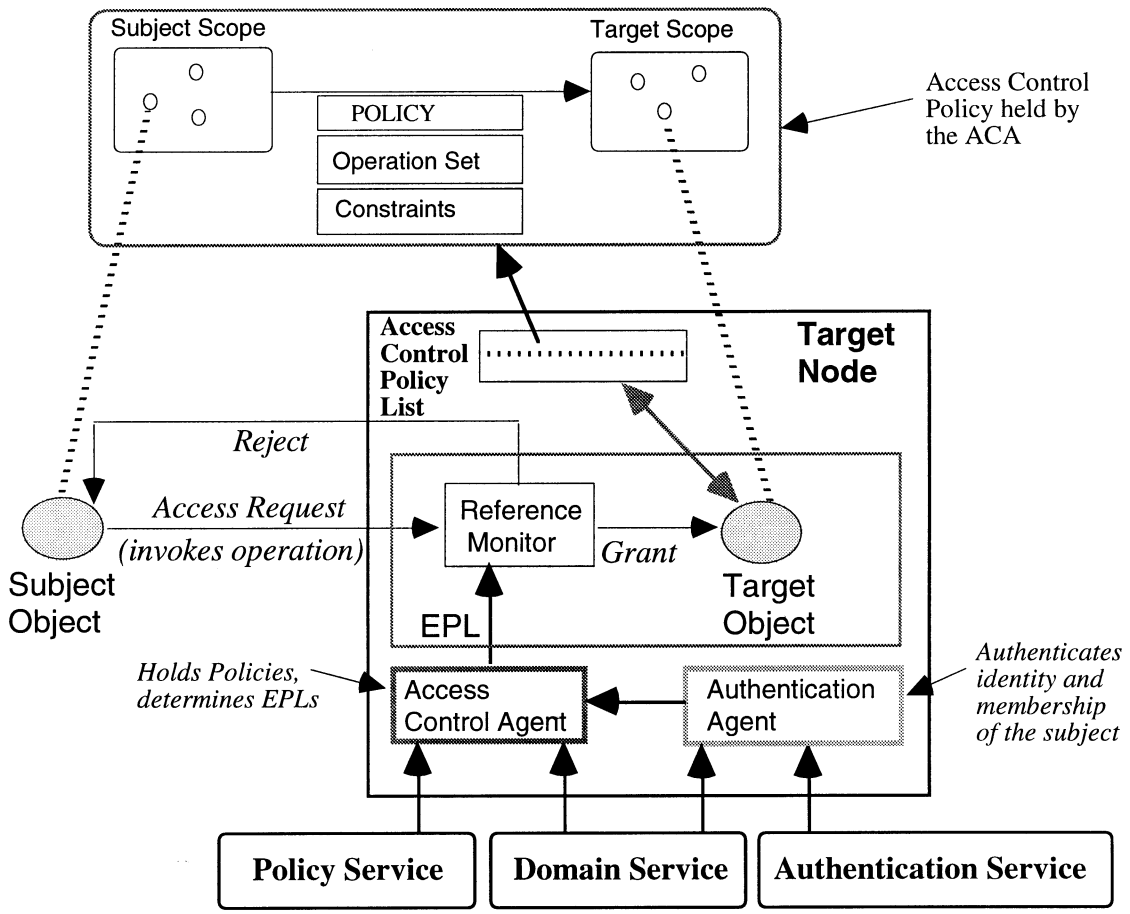


Figure 1.1 Access Control Policy Enforcement Model

The *Access Control Policy List (ACPL)* identifies all policies which apply to a particular target, and the *Enabled Policy List (EPL)* is the subset of the APCL permitting a particular authenticated subject to access the target. The ACPL and EPL are determined by the ACA.

[1]This operator cannot be used in the subject scope expression as the authentication system cannot support verification of non-membership in domains. In distributed systems, in general, it is impossible to prove non membership. For a short discussion on this issue see [Abadi *et al.* 1993].

There is one *Reference Monitor (RM)* supporting all objects within an address space in a node; it makes access control decisions on whether or not an operation is permitted, based upon the EPLs which are given to it by the ACA. The access control mechanism is described in detail in [Yialelis *et al.* 1995]. The authentication agent supports authentication of domain membership and individual object identities on behalf of all objects in a node (see section 2 for more detail).

The emphasis of this paper is on the basic design of the authentication system that supports domain membership authentication. Section 2 presents the basic architecture of the authentication service and explains how secure channels between remote objects are established using a trusted authentication service. Section 3 shows how this authentication service can be used to authenticate domain membership and cope with frequently changing domain structures. Finally, section 4 summarises the design and discusses further work. A list of the abbreviations used in this paper is given in the appendix.

2. ARCHITECTURE OF THE AUTHENTICATION SERVICE

The efficiency of the authentication mechanism is critical as a subject may be member of multiple domains requiring the authentication of many membership certificates. The overheads of generation and verification of membership certificates based on the use of public-key encryption would be too time consuming. Symmetric encryption based on shared secret keys is used in our system as it is about 1000-5000 times faster than public key encryption [Lampson *et al.* 1992].

The *Authentication Service* (AS) is provided by a number of replicated *Authentication Servers*. An *Authentication Agent* (AA) in each host interacts with the AS to perform identity and domain membership authentication. The utilisation of the AAs makes authentication transparent to the application objects. All AAs are registered with the AS during the bootstrapping of their hosts.

2.1 Replicated Authentication Servers

Authentication systems based on secret-key cryptography use a trusted on-line authentication service which is usually provided by replicated authentication servers to improve both performance and availability (e.g. Kerberos [Stallings 1995]). These servers hold identities and secret keys of users and servers so it is difficult to maintain security and consistency of this state information held by these replicas. In order to avoid these problems we have adopted a scheme similar to the one described in [Davis *et al.* 1990] which utilises replicated servers with minimal state (figure 2.1). The advantages are that very simple tamper proof machines can be used as authentication servers, and replication permits performance improvement without problems of maintaining consistency (see also [Lampson *et al.* 1992]).

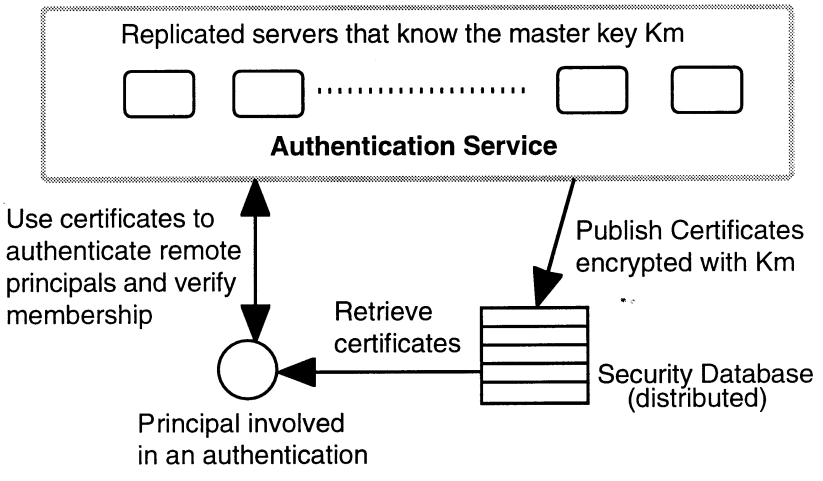


Figure 2.1 Using Private Key Certificates

The security database shown in figure 2.1 consists of *private-key certificates* containing encryption keys and identities for the principals that have been registered to the AS i.e. authentication agents and users. These certificates are encrypted using a *master key* known only to the replicated authentication servers [Davis *et al.* 1990] to provide both secrecy and integrity. The principals retrieve the certificates from the security database which need not be secure i.e. these are similar to public key certificates in an asymmetric cryptosystem. The master key is the only state information held by authentication servers which has to be kept very secure. The replicated authentication servers can also be used as translators (relays) [Lampson *et al.* 1992]. That is, they decrypt received messages from the sender and re-encrypt them with the secret key of the receiver.

In this paper we assume that the clocks of the authentication servers and the other nodes of the system are *loosely synchronised.* If this assumption cannot be made, the protocols presented in the following sections can be changed to use the challenge-response approach though this will increase the number of communication steps required.

Usually, a set of replicated authentication servers can only be trusted by a limited number of principals; for example by the principals in a university department or an organisation. We use the term *realm* (as in the Kerberos model) to denote the set of principals that are served by a certain collection of replicated authentication servers. This report discusses intra-realm authentication, i.e. when the principals involved in an authentication belong to the same realm. Inter-realm authentication is also possible by registering ASs in a Certification Authority (CA) hierarchy [Lampson *et al.* 1992] .

2.2 Notation

This section discusses the notation used to represent encrypted messages, private-key certificates and statements. A message M encrypted under the secret key K is represented as $\{M\}_K$. We assume that a secret-key cryptosystem that provides both *secrecy* and *privacy* is used.

The expression $A \Rightarrow B$ denotes that A *speaks for* B where A and B are principals. The relation *speaks for* is defined as in [Lampson *et al.* 1992], that is A *speaks for* B if the fact that principal A says something means that we can believe that principal B says the same thing. The operator $\Rightarrow$ is transitive which means that if $A \Rightarrow B$ and $B \Rightarrow C$, then $A \Rightarrow C$. If secret keys are considered to be principals, the statement $K \Rightarrow B$ means that if we receive a message M encrypted under the key K, we can believe that M has been said by B.

The notation $A \xleftrightarrow{K} B$ is used as in [Burrows *et al.* 1990], to denote that the shared key K can be used by the principals A and B to communicate with secrecy and integrity. In the interests of clarity, we also use this notation in the description of the authentication protocols. If A believes that $A \xleftrightarrow{Kab} B$ is true, it derives that $Kab \Rightarrow B$. Similarly, B derives $Kab \Rightarrow A$ if it believes that $A \xleftrightarrow{Kab} B$ is true. In addition, if both A and B believe $A \xleftrightarrow{Kab} B$, we say that a *secure channel* has been established between principals A and B.

Each private-key certificate encrypted by an authentication server contains the *id* of the server and the time it was generated which form the unique id of the certificate (CID). In addition, each certificate contains an *expiration time*, when the certificate becomes invalid. The format of a private-key certificates is:

$$\{CID, Te, < statement >\}_{Km}$$

where Km is the master key of the AS and Te is the expiration time. In the interests of brevity, we usually omit the CID or the expiration time in examples.

2.3 Secure Channels between Authentication Agents

We assume that a secure communication channel exists between any two objects on the same node as they communicate via the system software/hardware which has to be trusted anyway. It is necessary to provide means of establishing secure communication channels between objects in different nodes. In general, these channels should be able to provide both secrecy and integrity. The establishment of object-to-object secure channels is undertaken by the node AAs and it requires a secure channel between them.

When an AA of node *X* (*AAx*) is registered to the AS a secure channel (that provides both secrecy and integrity) is established between the AS and *AAx* (see Figure 2.2). The registration of the AA has to be done by the security administrator who has the right to register AAs to the AS. The shared secret key *Kx* used for this channel (see also in [Davis *et al.* 1990] a discussion on key distribution) is certified by a private-key certificate of the form:

$$\{CID, Te, AAx \xleftrightarrow{Kx} AS\}_{Km}$$

We assume here that the transmission of *Kx* to *AAx* takes place off-line as at that moment there is no on-line channel that provides secrecy and integrity between the AS and the AAx.

Including the above-mentioned certificate in messages from *AAx* to AS, AAx and AS are able to use the established channel even though AS does not store *Kx*, as this key can be retrieved from that certificate.

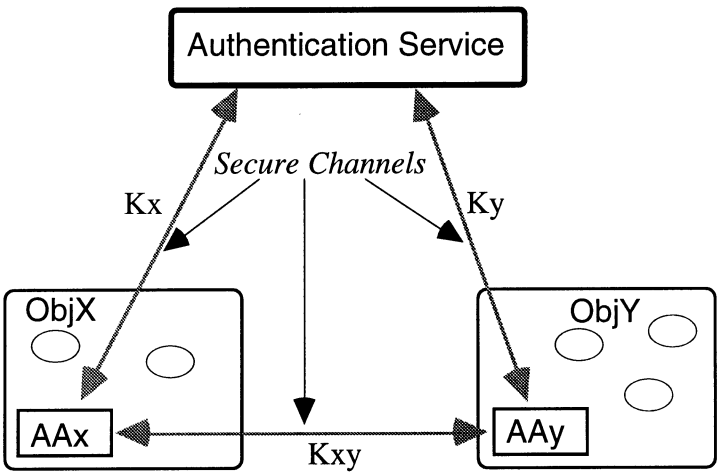


Figure 2.2 Secure channels between AAs, and between AAs and AS

The following protocol is used to establish a secure channel between two authentication agents *AAx* and *AAy*.

Message 1. AAy -> AAx: $\{Te_2, AAy \xleftrightarrow{Ky} AS\}_{Km}$

Message 2. AAx -> AS: $\{Te_1, AAx \xleftrightarrow{Kx} AS\}_{Km}, \{Te_2, AAy \xleftrightarrow{Ky} AS\}_{Km}$

Message 3. AS -> AAx: $\{Ts, Te_3, AAx \xleftrightarrow{Kxy} AAy\}_{Kx}$, $\{Ts, Te_3, AAx \xleftrightarrow{Kxy} AAy\}_{Ky}$

Message 4. AAx -> AAy: $\{Ts, Te_3, AAx \xleftrightarrow{Kxy} AAy\}_{Ky}$, $\{AAx; Tx\}_{Kxy}$

Message 5. AAy -> AAx: $\{Tx\}_{Kxy}$

This protocol is similar to the one employed by Kerberos which is analysed in [Burrows *et al.* 1990]. The main difference is that the authentication server in our system is provided with two private-key certificates that mention the keys it shares with the AAs involved in the protocol. In the first message, *AAx* gets the private-key certificate that mentions the secret key of *AAy*. In message 2, *AAx* sends that certificate along with the one that mentions its own secret key Kx. At

this point the authentication server involved in the protocol can decrypt these two certificates as it knows the master key *Km*. If we assume that no revocation of certificates takes place[2], we can state that:

$$AS \textbf{ believes } AAx \overset{Kx}{\longleftrightarrow} AS \text{ and } AS \textbf{ believes } AAy \overset{Ky}{\longleftrightarrow} AS$$

The rest of the protocol can now be analysed using BAN logic as in [Burrows *et al.* 1990] Upon the completion of the protocol we derive that:

$$AAx \textbf{ believes } AAx \overset{Kxy}{\longleftrightarrow} AAy \qquad AAx \textbf{ believes } AAy \textbf{ believes } AAx \overset{Kxy}{\longleftrightarrow} AAy$$

and

$$AAy \textbf{ believes } AAx \overset{Kxy}{\longleftrightarrow} AAy \qquad AAy \textbf{ believes } AAx \textbf{ believes } AAx \overset{Kxy}{\longleftrightarrow} AAy$$

The shared key *Kxy* can then be used for secure communication between *AAx* and *AAy*. If, for instance, *AAx* sends a time stamped message *M* encrypted with *Kxy*, i.e. $\{Ts, M\}_{Kxy}$, *AAy* derives that *AAx* **says** *M*. This is because $Kxy \Rightarrow AAx$. The timestamp *Ts* guarantees the freshness of the message (i.e. the message is not a replay of an old message). In addition, as no untrusted principal can see the key *Kxy*, we also achieve message secrecy.

In fact, as soon as a shared secret key has been established between two AAs, further session keys can be established without the use of the AS. These keys can either be used for secure communication between the AAs or for secure communication between the objects on the two nodes. The second use is explained in the following section.

2.4 Channels between Objects

AAs establish secure channels on behalf of the objects on their nodes. In theory, a single key can be used for more than one object-to-object channel between two nodes provided that this secret key is not revealed to the objects[3] and it is only used for encryption of a limited amount of traffic. However, not all channels between objects require the same type or level of security[4]. This indicates that different object-to-object channels should use different keys (which may correspond to different cryptosystems). For instance, in figure 2.2, *AAx* and *AAy* can establish a channel on behalf of *ObjX* and *ObjY* by choosing a shared secret key *Ks*. This key is given to *ObjX* and *ObjY* which can now selectively encrypt and decrypt the messages they send to each other. The effect is that the authentication and the establishment of the channel is transparent to the objects that use the channel.

An AA should also be able to verify that a remote authenticated AA is trusted to act on behalf of a given remote object. This is achieved by checking that the IP address of the AA (obtained from the OID) is the same as that of the object for which it acts. All OIDs of objects that reside on a node contain the same IP address even if the node has been allocated more than one IP address. It is implied here that the user that creates an object on a node trusts the AA of that node to support the object in terms of authentication and secure communication. This mechanism, however, cannot be used when migration of objects is expected to take place.

The establishment of a secure channel between two objects is triggered by the subject. A simplified procedure for channel establishment is illustrated in figure 2.3. In phase (1), the subject requests the establishment of a secure channel with the target. This request can also specify the type and level of security required. In phase (2), the two responsible AAs agree[5] on the key *Ks* that the two objects should use and on the channel identifier (CHID). In phase (3), the two partners of the channels are informed about the key and CHID of the new channel.

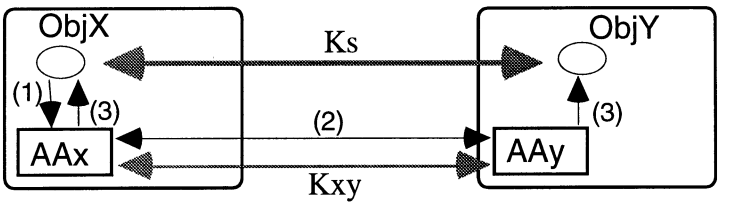
---

[2]Revocation is discussed in section 2.5.

[3]Objects on a node cannot, in general, be trusted to the same extent as the AA.

[4] For many applications the provision of integrity is sufficient. Provision of secrecy may not be required for efficiency or legal reasons.

[5]The two AAs need a secure channel to communicate. If such a channel does not exist, one has to be established as described in section 2.3.



Phase (1):
ObjX->AAx: EstablishChannel(ObjY, Channel_Type)
Phase(2):
AAx and AAy agree on Ks and CHID
Phase(3):
AAx->ObjX: Ks, CHID
AAy->ObjY: Ks, CHID

Figure 2.3 Establishing an Object-to-Object Secure Channel

The two AAs involved in the establishment of a channel maintain information related to the channel. Since a channel is associated with a subject and target, a CHID is a reference to the OIDs of these objects, to the shared secret key and cryptosystem they are using, and permits the reference monitor to identify the EPL associated with the subject. Because of the importance of domain membership for access control decision, the notion of the channel is augmented to represent the authenticated domain membership of the involved objects. Section 3 describes how the two authentication agents verify the domain membership of the objects that use a channel. The concept of the secure channel is discribed in detail in [Yialelis *et al.* 1995].

The encryption and decryption of the messages can be performed in the address space of the object that use a channel or in a separate address space that is trusted at least to the same extent as the object.

2.5 Certificate Revocation

Revocation of private-key certificates is more difficult with stateless authentication servers but we can move the task of rejecting revoked certificates from the authentication servers to the AAs. The AAs base their beliefs on statements issued by the AS. The correctness of these statements depends on a number of private-key certificates which are processed by the AS. The AS can therefore associate each statement with the certificates on which it is based. The AA that receives a statement along with the identifiers of the certificates it is based on, can check whether the statement is valid provided that it has access to a *Certificate Revocation List* (CRL). One of the authentication servers, referred to as the *Revocation Server*, can be dedicated to the maintenance and publication of the CRL. A single server is adequate as we do not expect a high revocation rate and one of the other servers can take over the function if it fails. The revocation server periodically publishes a CRL containing the CIDs of the certificates that have been revoked but have not expired, plus a digest (*H(CRL)*) of the CRL. The digest is encrypted with the master key *Km*. An AA gets the latest CRL every time it runs an authentication protocol and checks its integrity by requesting translation of the encrypted digest from the AS (figure 2.4).

The protocol for establishment of secure channels has to be changed slightly in order to cope with revoked certificates. The statements that the AS generates should contain a Certificate Dependence List (CDL). This CDL contain the CIDs of the certificates on which the statement is based on. In the protocol given below, the CDL in the statement that is addressed to *AAx* is *(CID2)* while the CDL in the statement addressed to *AAy* is *(CID1)*. The two authentication agents should check these lists against the current version of the CRL in order to reject the statement if it is based on a revoked certificate:

Message 1. AAy -> AAx: $\{CID2, Te_2, AAy \xleftrightarrow{Ky} AS\}_{Km}$

Message 2. AAx -> AS: $\{CID1, Te_1, AAx \xleftrightarrow{Kx} AS\}_{Km}, \{CID2, Te_2, AAy \xleftrightarrow{Ky} AS\}_{Km}$

Message 3. AS -> AAx:

$$\{CDL1, Ts, L, AAx \xleftrightarrow{Kxy} AAy\}_{Kx}, \{CDL2, Ts, L, AAx \xleftrightarrow{Kxy} AAy\}_{Ky}$$

Message 4. AAx -> AAy: $\{CDL2, Ts, L, AAx \xleftrightarrow{Kxy} AAy\}_{Ky}$, $\{AAx, Tx\}_{Kxy}$

Message 5. AAy -> AAx: $\{Tx\}_{Kxy}$

The revocation server publishes a new version of the CRL when it receives a new revocation request. In addition, it publishes a new version of the CRL (even if it is empty or identical to the previous version) periodically for instance every hour. In this way, denial of service attacks can be detected and counter measures can be taken according to the security policy applied to the system.

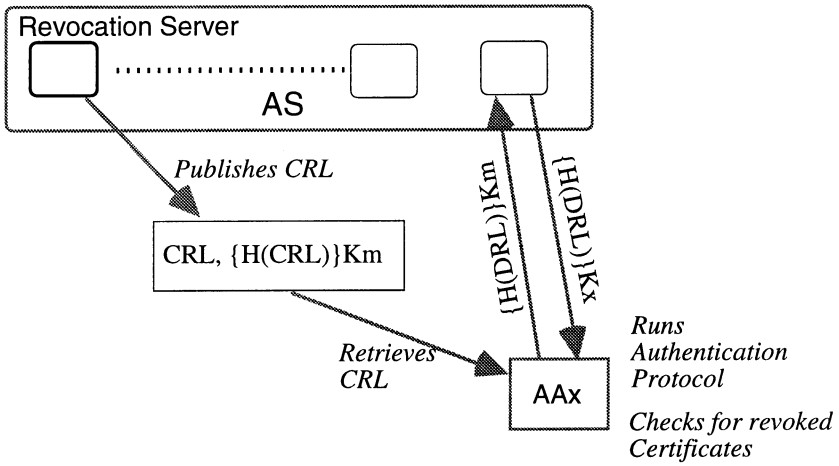


Figure 2.4 Revocation of Certificates

3. AUTHENTICATION OF DOMAIN MEMBERSHIP

We present a domain membership authentication scheme that is based on the assumption that the domain service is trusted to certify membership of objects. More specifically, we rely on the parent domains of the claimant in order to verify the membership it claims. Domains are distributed in server components which reside on manager workstations or on dedicated servers around the system. This imposes some difficulties in the design of the system since there is no single trusted server to provide information about the domain membership of the subjects (c.f. the notion of the Privilege Attribute Server-PAS in the SESAME architecture [Parker *et al.* 1994]). Membership authentication is achieved by utilising the AS as translator (relay). It re-encrypts membership statements with the secret key of the AA that has to verify the membership of the claimant. So we avoid the creation of secure channels between the domain servers that certify a membership and the AA of the verifier.

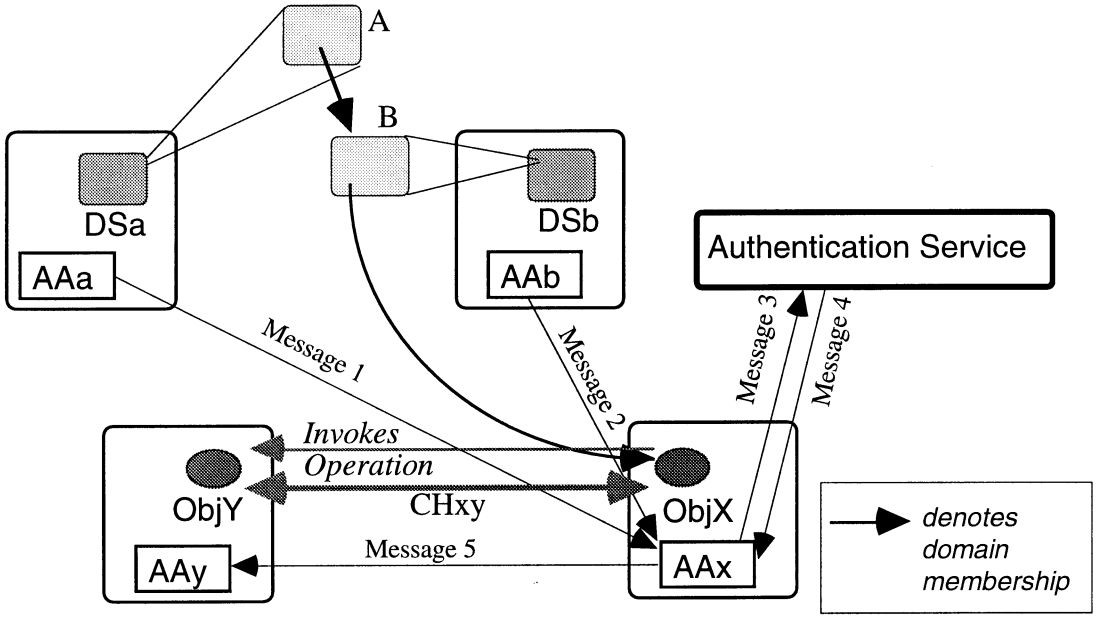
In this section, we denote the fact that an object A is *direct member* of the domain B as $A \prec B$. Similarly, if A is *indirect member* of B, we write $A \prec\prec B$.

3.1 A simple example

In the example illustrated in figure 3.1, we assume that *ObjX* has to be authenticated as indirect member of domain A by AAy. AAx is also involved in the authentication as it acts as representative of *ObjX*. *ObjX* is a direct member of the domain B which is maintained at the

domain server *DSb*. *B* is also a direct member of *A* which is maintained at the domain server *DSa*. In this case, there is no single domain server that can state $ObjX \prec\prec A$. Instead, the verifier (*AAy*) combines two individual statements $ObjX \prec B$ and $B \prec A$ uttered by the two domain servers in order to establish $ObjX \prec\prec A$. These statements are received by the verifier through a communication channel that ensures the integrity of the messages. One way to achieve this is to establish two secure channels between the verifier and the two domain servers as we have explained in section 2.4. This is not efficient for transmitting a single message, so we have adopted a mechanism which is based on the ability of the AS to translate (re-encrypt) messages.

According to this mechanism, *AAx* (which acts on behalf of *ObjX*) collects two membership statements $\{Ts1, A \textbf{ says } (B \prec A)\}_{Ka}$ and $\{Ts2, B \textbf{ says } (ObjX \prec B)\}_{Kb}$ which have been encrypted with the secret keys of the AAs of their respective domain servers. These statements cannot be read by the verifier (*AAy*) but they can be re-encrypted with the secret key of *AAy* by the AS. In fact, the authentication server that receives these encrypted statements establishes AAa **says** $(A$ **says** $(B \prec A))$ and AAb **says** $(B$ **says** $(ObjX \prec B))$ since $Ka \Rightarrow AAa$ and $Kb \Rightarrow AAb$. If $AAa \Rightarrow A$ and $AAb \Rightarrow B$ hold, the server also establishes that A **says** $(B \prec A)$ and B **says** $(ObjX \prec B)$. The server goes on to encrypt two statements that can be read by *AAy*: $\{Ts1, A \textbf{ says } (B \prec A)\}_{Ky}$ and $\{Ts2, B \textbf{ says } (ObjX \prec B)\}_{Ky}$. These statements are sent to *AAx* which forwards them to *AAy*. *AAy* can now decrypt these statements and establishes that A **says** $(B \prec A)$ and B **says** $(ObjX \prec B)$ (*AAy* believes that $Ky \Rightarrow AS$ and that AS is trusted on its statements). So *AAy* eventually establishes $ObjX \prec\prec A$. The complete protocol sequence for this example is given in figure 3.1.



Message 1. AAa -> AAx: $\{Ts1, A \textbf{ says } (B \prec A)\}_{Ka}$, $\{Te1, AAa \overset{Ka}{\longleftrightarrow} AS\}_{Km}$

Message 2. AAb -> AAx: $\{Ts2, B \textbf{ says } (ObjX \prec B)\}_{Kb}$, $\{Te2, AAb \overset{Kb}{\longleftrightarrow} AS\}_{Km}$

Message 3. AAx -> AS: $\{Ts1, A \textbf{ says } (B \prec A)\}_{Ka}$, $\{Te1, AAa \overset{Ka}{\longleftrightarrow} AS\}_{Km}$

$\{Ts2, B \textbf{ says } (ObjX \prec B)\}_{Kb}$, $\{Te2, AAb \overset{Kb}{\longleftrightarrow} AS\}_{Km}$,

$\{Te3, AAy \overset{Ky}{\longleftrightarrow} AS\}_{Km}$

Message 4. AS -> AAx: $\{Ts1, A \textbf{ says } (B \prec A)\}_{Ky}$, $\{Ts2, B \textbf{ says } (ObjX \prec B)\}_{Ky}$

Message 5. AAx -> AAy: $\{Ts1, A \textbf{ says } (B \prec A)\}_{Ky}$, $\{Ts2, B \textbf{ says } (ObjX \prec B)\}_{Ky}$

Figure 3.1 Example of Domain Membership Authentication

The method illustrated in this example can be generalised for authentication of multiple levels of indirect membership. The procedure is repeated for each of the membership to be authenticated.

3.2 Channels and Domain Membership

In our system, authentication of domain membership is performed in the framework of an established secure channel between a subject and a target object. In the example mentioned in section 3.1, the authenticated membership $ObjX \prec\prec A$ is associated with the channel CHxy between *ObjX* (subject) and *ObjY* (target). The authenticated membership of the subject is used by the ACA of the target to generate the EPL. The RM can use the CHID identified in a request to determine which EPL to check.

The subject of a channel may be a direct or indirect member of a large number of domains, but only the membership that satisfies the subject scopes of the access control policies, is used by the ACA to generate an EPL . The decision as to which membership should be verified is made by the access control service [Yialelis *et al.* 1995].

In addition, the subject may request the authentication of domain membership of the target of a channel. In this case the subject triggers the authentication and the responsible AAs swap roles; that is the AA of the subject becomes the verifier while the AA of the target collects the necessary statements.

Note that the encrypted messages in figure 3.1 contain timestamps which are used to verify the freshness of the membership statements. When a membership statement expires while the channel that has been associated with is still in use, re-authentication is required by the AA that acts as verifier. If the re-authentication fails, the channel becomes invalid. In this way the system copes with changing domain membership. The lifetime of the membership statements can be adjusted to comply with the security and efficiency requirements of the system.

4. CONCLUSIONS AND FURTHER WORK

We have outlined the design of an authentication service for object based systems that makes use of the notion of domains as a means of specifying policies in terms of groups of objects to cope with large scale systems that contain millions of objects. Enforcement of policies that have been specified in terms of domains require authentication of object membership of domains.

The described system is based on symmetric cryptography and utilises an authentication service which is provided by servers holding minimal state, i.e. only the master key. This makes replication of secure servers easy as there are no consistency problems to be overcome. In addition, the authentication service can be used as a translator (relay) in order to avoid time consuming establishment of secure channels when authentication of domain membership is performed.

There is a small Trusted Computing Base, i.e. an authentication agent (AA) and access control agent (ACA), which is replicated on every node in the system. Only the AAs and the users need to be registered with the AS and the authentication mechanisms are transparent to other objects.

An important issue which has not been discussed in this paper is the delegation *of access rights*. The design supports cascaded delegation of access rights by utilising the authentication service as translator for the *delegation tokens* which prove that a delegation has taken place. The notion of the channel is augmented to support delegation by associating it with delegated access rights and the OIDs of the objects that have delegated access rights. Revocation of delegation is supported by the revocation server which periodically issues a list that contains the ids of the revoked delegation tokens.

The design also supports user authentication which can be achieved either by following the conventional password paradigm or by using smart cards. We are working on the design of

inter-realm authentication which registers individual authentication services within a CA-hierarchy.

The domain service has been fully implemented and the authentication service and access control system are being implemented within the SysMan project.

Acknowledgements
We gratefully acknowledge financial support from the Swiss Bank Corporation (London), and Esprit SysMan (7026) and IDSM (6311) projects. We also acknowledge the contribution of our colleagues, working on these projects, to the concepts discussed in this paper.

REFERENCES

[Abadi *et al.* 1993] M. Abadi, M. Burrows, B. Lampson and G. Plotkin, "A Calculus for Access Control in Distributed Systems", *ACM Transactions on Programming Languages and Systems*, Vol. 15(4), pp. 706-734, 1993.

[Burrows *et al.* 1990] M. Burrows, M. Abadi and R. Needham, "A Logic of Authentication", *ACM Transactions on Computer Systems*, Vol. 8(1), pp. 18-36, 1990.

[Davis *et al.* 1990] D. Davis and R. Swick, "Network Security via Private-Key Certificates", *ACM SIGOS Operating Systems Review*, Vol. 24(4), pp. 64-67, 1990.

[Lampson *et al.* 1992] B. Lampson, M. Abadi, M. Burrows and E. Wobber, "Authentication in Distributed Systems: Theory and Practice", *ACM TOCS* Vol. 10(4), pp. 265-310, 1992.

[Parker *et al.* 1994] T. Parker and D. Pinkas, "SESAME V2 - Overview", SEASAME, 1994.

[Sloman et. al 1993] M. Sloman, J. Magee , K. Twidle, and J. Kramer, An Architecture for Managing Distributed Systems, *Proc. 4th IEEE Workshop on Future Trends of Distributed Computing Systems,* Lisbon, Sep. 1993, pp 40-46.

[Sloman 1994] M. Sloman, Policy Driven Management for Distributed Systems, Plenum Press *Journal of Network and Systems Management*, Vol.2(4), pp. 333-361, 1994.

[Stallings 1995] W. Stallings, Network and Internetwork Security, Prentice Hall, 1995.

[Yialelis *et al.* 1995] Nicholas Yialelis and Morris Sloman, "A Security Framework Supporting Domain Based Access Control in Distributed Systems", Research Report DoC 95/14, Imperial College, 1995.

APPENDIX

The following abbreviations are used in this paper:

AA	Authentication Agent
ACPL	Access Control Policy List
AS	Authentication Service
CA	Certification Authority
CDL	Certificate Dependence List
CHID	Channel Identifier
CID	Certificate unique Identifier
CRL	Certificate Revocation List
EPL	Enabled Policy List
OID	Object Identifier
RM	Reference Monitor
TCB	Trusted Computing Base
UID	object Unique Identifier