

A Distributed Credential Management System for SPKI-based Delegation Systems

Óscar Cánovas[†], Antonio F. Gómez[‡]

[†]*Department of Computer Engineering*

[‡]*Department of Information and Communications Engineering*

University of Murcia

30071 Murcia (Spain)

ocanovas@um.es, skarmeta@dif.um.es

Abstract

Traditionally, certificates have been used to link a public key to a particular name identifying that key. However, public key certificates are digitally-signed statements which can be used in order to assert many other types of information. SPKI has become one of the most outstanding proposals referring to authorization, and several applications have been based on SPKI certificates in order to provide authorization services to well-known scenarios in distributed systems. Most of these scenarios are based on delegation, where resource guards have an ACL with few entries granting keys belonging to some authorization or naming authorities the right to delegate all access to the controlled resources. These authorities can issue certificates delegating these permissions to other subordinates authorities, or to specific users. In this way, the structure generated reflects the system management process. However, generation of these certificates usually is system-dependent. In this paper, we present a management system that can be used in all SPKI scenarios based on delegation. This system addresses some problems related to scalability, certificate distribution, and interoperability. We define how certification requests can be expressed, how different security policies can be enforced using this system, which are the entities involved in a certification scenario, and we propose a mechanism able to exchange authorization-related information among these entities.

1 Introduction

Loren Kohnfelder defined "certificate" in 1978 as a digitally-signed statement holding a name and a public key, and nowadays the words *certificate* and

identity certificate are still used as synonyms. However, a certificate is a record stating some information about the entity the certificate was issued to, and this information may be a role membership statement, or an authorization. Authorization certificates bind a capability to a key, and this capability can be used to determine what the entities are allowed to do.

One of the most outstanding proposals related to this type of certificates has been the SPKI/SDSI infrastructure [8]. SPKI/SDSI provides three types of digital certificates (ID, attribute, and authorization) that can be used in several security scenarios. In fact, there are several proposals which make use of SPKI certificates in order to provide authorization services to many different application environments, such as WLAN networks [10], CORBA distributed objects [12], or web servers [4]. Most of these scenarios are based on delegation, where resource guards have an ACL with few entries granting keys belonging to some authorization or naming authorities the right to delegate all access to the controlled resources. However, some of these proposals do not explain how certificates are issued by the authorities, and this usually is application-dependent. Although simple and not distributed approaches can constitute a good alternative for small scenarios, some problems derived from scalability or interoperability might arise in more complex environments [3]. Generation or revocation of these certificates should not be implemented using simple command-line applications. A structured and distributed system must be provided.

A system is necessary which addresses the problems related to scalability, certificate distribution, and interoperability. In this paper, we present

DCMS (Distributed Credential Management System). DCMS defines how certification requests should be expressed, how different security policies can be enforced using this system, which are the entities involved in a certification scenario, and how these entities can exchange authorization-related information. We have used the AMBAR (Access Management Based on Authorization Reduction) protocol [2] in order to perform that exchange, but similar protocols can be also used. This system is divided into the naming management system (NMS), which manages the issues related to SPKI ID certificates, and the authorization management system (AMS), which is responsible for those procedures related to SPKI attribute and authorization certificates. We believe that this system can lead up to the definition of an application-independent system which can be used in order to provide authorization services to many different scenarios based on delegation. DCMS also complements some proposed mechanisms for revocation and validation of SPKI certificates [11], and can make use of public repositories for certificate storage purposes [9].

We can find similar proposals in the literature. In [13], a security architecture is presented which is related to authentication, authorization and delegation in a distributed environment based on SPKI. This proposal differs from DCMS about object formats, and system structure. We use *s*-expressions in order to specify the authorization policies and requests, instead of HTTP-like messages and codes. We do not find necessary to use a different encoding since SPKI-like *s*-expressions are appropriate, straightforward, and standard. Moreover, we do provide a generic framework of authorities, proxies and protocols that can be used as guidelines to design and implement authorization management services. In fact, our system has been implemented using the Intel version 3.14 of CDSA (Common Data Security Architecture) [5].

This paper is organized as follows. Section 2 presents an authorization scenario based on delegation in order to clarify why DCMS is useful. Section 3 provides some basic background on the AMBAR protocol. Section 4 presents the entities involved in the naming management system (NMS), and shows the *s*-expressions that will be used in this system to specify certification requests and access control lists. Section 5 contains similar details concerning the authorization management system (AMS). Section 6 presents how system entities can interoperate using the AMBAR protocol. Finally, Section 7

makes some concluding remarks.

2 Motivation

In this section we are going to show a distributed system where SPKI certificates and delegation can be used to implement physical access control [3]. We will also explain why DCMS is necessary.

This distributed system is based on a RBAC (Role Based Access Control) model [15]. The central concept of RBAC is that permissions are associated with roles, and users are assigned to appropriate roles. This greatly simplifies management of permissions since the two relations are considered completely independent.

In this system, special devices named TICA are used, which are able to perform some access control operations like opening doors. They are located at the entrances of the different buildings and/or departments, and they can establish their own access control conditions, trusted entities, and authorization mechanisms. TICAs delegate authorization management to particular authorization authorities (AA). This is accomplished through authorization certificates issued by the TICAs for a set of specific AAs. These certificates basically give the AAs total authority over the device, and also permission to further delegate the access control is granted. TICAs can also delegate the authority by means of ACL entries containing the same information included in those certificates. Then, AAs usually create new attribute certificates giving a subset of permissions to the roles defined by any of the existing naming authorities (NA). Roles are managed by NAs, which issue ID certificates in order to state that a particular user has been assigned to a specific role. In this way, TICAs are the beginning of the authorization path, and not only the policy enforcement point. The device is able to make the security decision regarding the authorization data presented by the user requesting the access.

However, this certification management process must be designed and implemented using a scalable approach. An encoding for certification requests must be defined, and a mechanism is necessary which is able to exchange authorization-related information among the entities involved.

Using DCMS, once TICAs have delegated the authorization management task to the different au-

thorities, principals can request individual certificates in order to gain access. These requests can be generated and sent to the authorities by the principal itself, or can be submitted using trusted service access points (SAP). Authorities will issue the requested certificates depending on the authorization policy (authorities are the policy decision point). This policy can be represented using SPKI ACLs, a database or any other method, and it is system-dependent, although in the next sections we will assume that it is implemented using ACLs.

Figure 1 shows a particular scenario where TICAs delegate the authorization tasks to different AAs. Users make use of DCMS in order to obtain specific authorization certificates from these entities. In this case, we assume that authority *A* and the SAP are the AMBAR peers. Once the certificates are generated, these can be presented to the TICAs in order to gain access.

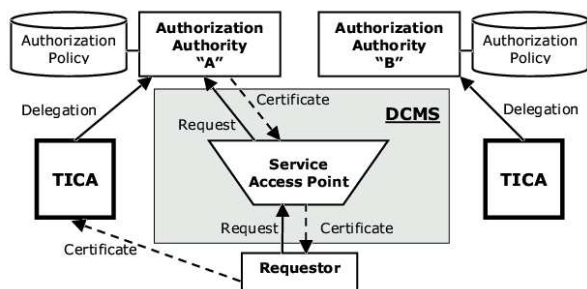


Figure 1: Use of DCMS

3 AMBAR Protocol

AMBAR (Access Management Based on Authorization Reduction) [2] is a protocol for secure exchange of authorization-related information based on public key cryptography. This protocol does not depend on a particular type of authorization or identity-based certificate, and it contains a negotiation phase designed to adapt the protocol to access control scenarios with different requirements (anonymity, confidentiality, credential recovery, etc.). In general, it provides functionality to transmit resource access requests, the authorization information related to those requests (credentials, ACLs), and results obtained from a certificate chain discovery method or compliance checker. ACLs can be transmitted in order to give some information to the client about which credentials would be necessary to access the resource. However, disclosure of security policies

(ACLs are particular implementations of these policies) must be carefully performed since they can contain sensitive information [16].

It has been designed to be session-oriented in order to optimize those scenarios where the request/response messages are exchanged between the same client and server. In addition, it does not need to rely on any additional security protocol since it adds confidentiality and integrity to the data being transmitted.

The AMBAR protocol consists of different components organized, as Figure 2 illustrates, in two layers.



Figure 2: AMBAR Architecture

- **Session Management module (SM).** This module transmits the client and server security preferences, and generates the cryptographic data used by the TC layer to protect the subsequent communications. Clients and servers negotiate the following parameters:
 - *Symmetric cipher.* Parties select the symmetric cipher and its key length.
 - *Operation mode.* AMBAR supports two operation modes: anonymous client mode and fully identified.
 - *Identity-based certificates.* It is possible to select X.509, OpenPGP, or SPKI certificates.
 - *Authorization-based certificates.* AMBAR supports SPKI certificates, PKIX attribute certificates and KeyNote asserts.
 - *Credentials distribution.* Parties can select whether the credentials will be provided by the client (push), or will be obtained by the server from either a repository or an issuer (pull).
- **Request Management module (RM).** The RM module transmits two types of messages: messages related to authorization requests and credentials; and messages related to decisions and ACLs. Contents and the sequence of these

messages are determined by the negotiated operation mode and the method for distribution of credentials. As we mentioned previously, a session-oriented protocol allows some optimization to be performed. Therefore, the RM module could be responsible for optimizing authorization computations.

- **Authorization Results Management module (ARM).** The ARM module generates notifications and transmits the demanded resources. Negative notifications are transmitted by the server when the access is denied. If the access were granted, there would be two possible response messages: an affirmative notification if the client requested the execution of remote actions; or the controlled resource. It also enables (disables) the DSM module when an authorization request demanding the establishment (conclusion) of a data stream is granted.
- **Error Management module (EM).** Systems use the EM module to signal an error or caution condition to the other party in their communication. The EM module transmits a severity level and an error description.
- **Data Stream Management module (DSM).** The described request/response model is not suitable if we plan to use AMBAR as a transparent layer providing confidentiality, authentication and access control services. The DSM module, initially disabled, controls the transmission of arbitrary data streams, which are enabled once a request demanding the activation of this module is granted.
- **Transport Convergence module (TC).** The TC module provides a common format to frame SM, RM, ARM, EM, and DSM messages. This module takes the messages to be transmitted, authenticates the contents, then applies the agreed symmetric cipher (always a block-cipher), and encapsulates the results. The cryptographic data used to protect the information is computed by the SM module during the negotiation phase.

The AMBAR protocol is part of a complete authorization framework for certificate-based access control systems. It is implemented with the Intel 3.14 version of CDSA (Common Data Security Architecture) [5]. We have used the CSP (Cryptographic

Service Provider) module built upon OpenSSL, and the X.509 and SPKI CL (Certificate Library) modules. We decided to use CDSA since this architecture provides all security services necessary to implement the framework and additionally, this provides integrity services which can be used to ensure component integrity and trusted identification of the component's source.

4 Naming Management System (NMS)

As we mentioned previously, DCMS is composed by two subsystems, NMS and AMS. In this section we are going to present the naming management system, which is responsible for the certification operations related to SPKI ID certificates. This type of certificates can be used to link a name to a particular principal (public key), and also to define group membership. NMS is very useful when authorization is based on group membership. In relation to the scenario presented in Section 2, we can imagine a TICA granting physical access to those principals which are members of group G . NMS can be used by principals in order to obtain an ID certificate for group G , which is issued by a particular naming authority.

Naming is not a requirement of distributed systems, but it is worth noting that large-scale SPKI-based delegation systems can be simplified using this mechanism. Naming is an optional tool for group management which can be useful to address scalability of complex systems.

4.1 Architectural elements

Figure 3 shows the three types of entities involved in NMS: requestors, service access points, and naming authorities. In this section we are going to give a brief description about these core entities, we introduce why they are necessary and how they interoperate.

- **Requestor.** A requestor is a principal demanding the generation of a new ID certificate. This entity must create a certification request and must send it to a particular naming authority (NA) in order to obtain the demanded certificate. This submission can be accomplished using a service access point or making use of an AMBAR connection between the

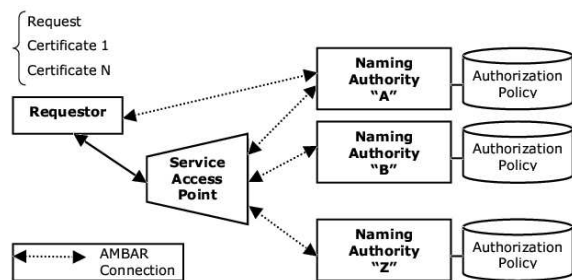


Figure 3: NMS entities

requestor and the NA. Other certificates can be attached to the request in order to demonstrate that the principal has permission to obtain the demanded certificate. There are two types of requestors: first, the principal demanding an ID certificate for a particular public key; second, the principal demanding an ID certificate for a particular name (e.g. a certificate stating that group B is a subgroup of group A). As we will see later, these two situations are managed following different approaches.

- Service access point.** Requestors can make use of access points in order to submit their certification requests to the appropriate naming authorities. Access points are optional, but they are very useful since they provide several additional services to requestors. First, naming authorities can be hidden from users. Moreover, in some scenarios with many authorities, it might be complicated to know which are the appropriate naming authorities for a particular ID certificate (especially with group membership certificates). SAPs can learn that location information from digitally-signed statements containing information about the system structure and properties. It is simpler to distribute this type of information to few SAPs than to all the principals. Finally, they can provide a certification service to requestors without AMBAR capabilities. Communication between requestors and access points is system-dependent, and it ranges from secure connections to public terminals placed at buildings or departments.
- Naming authority.** Naming authorities are the certificate issuers. They create ID certificates upon the requests received through the access points or directly from the requestors. NAs are controlled by a particular authorization policy, which can be implemented using

SPKI ACLs or other mechanisms. Whenever a NA receives a request and its related certificates, it executes a certificate chain discovery algorithm [6] in order to determine whether the certification request must be granted or denied. Inputs to this algorithm are the request, the additional certificates, and ACL entries. If a certificate chain is discovered, the algorithm returns the information that will be used to generate the new certificate. Communication with NAs are performed using AMBAR. As we have previously mentioned, AMBAR provides functionality to exchange authorization-related information. Using this protocol, entities can be authenticated (identification of requestors is optional), messages are encrypted and authenticated, and some optimization can be performed in order to avoid unnecessary calculations and transmissions (previous messages and authorization decisions can simplify further requests).

4.2 S-expressions for certification requests and ACL entries

Certification requests for ID certificates must contain information about the issuer defining the name, the name itself, the intended subject, and validity dates. Encoding can be based on s-expressions [14] since there is no need for making use of new syntax, and this can simplify the authorization process. Thus, requests might be encoded according to the representation form recommended by SPKI for the *authorization tag* field [7]. However, it is worth noting that the data elements contained in a request are also contained in a SPKI ID certificate, and therefore the structure for this type of certificates can be used. It is not necessary to define a completely new structure in order to express certification requests. Moreover, as we will explain, the same structure can be used by ACLs in order to encode authorization policies. S-expressions that we have used for certification requests and ACL entries have the following format:

```
(cert-request
  (issuer (name  $NA_i$   $N_i^j$ ))
  (subject  $P$ )
  (valid ..)
)
```

- cert-request.* This identifies the s-expression as a certification request.

- NA_i . This is the public key of the naming authority. This authority is responsible for issuing the ID certificates related to the name N_i^j .
- N_i^j . N_i^j is one of the names defined in the namespace of the authority NA_i .
- P . This is the principal (or principals) requesting the ID certificate. P might be:
 - A public key.
 - A set of entities. There are two possibilities in order to express a set of entities. On the one hand, we can use a group name, i.e., (`name NA N`). On the other hand, we can use the *-operator *set*, such as for instance (`* set Q R`), where Q and R must be public keys or names.
- *valid*. This specifies the requested validity period. The structure of this field is the one included in the SPKI standard.

If this s-expression is used as a certification request, P can only be a public key or a name, and it means that a new ID certificate is being demanded, whose issuer will be NA_i , P will be the subject, N_i^j will be the name linked to P , and will be valid during, at most, the specified validity interval. However, if this s-expression is included in the *tag* field of a SPKI-like ACL entry, it means that the principal (or principals) P are authorized to obtain an ID certificate from NA_i , where the name N_i^j will be linked to P (or each of the principals contained in P) during the specified validity period. Furthermore, N_i^j can make reference to several names when a (** prefix*) form or a (** set*) form is used.

Certification requests are encoded as sequences of two elements. The first element is the s-expression specifying the request, and the second one is a digital signature of that sequence. Signatures are encoded using the *signature* structure defined in [7], and they are generated using the requestor's private key. Requests have similar structure to certificates, but certificates are signed by issuers and requests are signed by requestors.

4.3 Some examples

In order to clarify how NMS entities cooperate to generate ID certificates, in this section we are going to analyze two certification requests. First, we

explain how a principal can obtain an ID certificate. Then, we will show how subgroups can be defined using ID certificates whose *subject* field also is a name. In these examples, authorization policies are represented by ACLs.

4.3.1 ID certificates for principals

In this first example, P is a principal demanding an ID certificate stating P as a member of group N_i^j , which is defined by NA_i . P creates the next certification request:

```
(sequence
  (cert-request
    (issuer (name  $NA_i$   $N_i^j$ ))
    (subject  $P$ ))
  (signature ..)
)
```

This request is sent to NA_i in order to obtain the demanded certificate. The request will be granted if NA_i can find a certificate chain from its ACL entries to the requestor's public key. The authority contains the next ACL:

```
(acl
  (entry
    (subject (name  $NA_l$   $N_l^k$ ))
    (tag (cert-request
      (issuer (name  $NA_i$   $N_i^j$ ))
      (subject (* set  $P$   $Q$   $R$ )))
    ))
  )
)
```

This ACL specifies that only members of N_l^k can request an ID certificate for N_i^j . If P , Q , or R were members of N_l^k they could request their own certificates. Otherwise, N_l^k can be considered as a relaying party able to make the request. In this case, we will assume that P is a member of N_l^k , and therefore P must send the next ID certificate in order to be authorized:

```
(cert
  (issuer (name  $NA_l$   $N_l^k$ ))
  (subject  $P$ )
)
```

Finally, the naming authority uses the data obtained from the authorization decision in order to create the certificate (signature has been omitted).

```
(cert
  (issuer (name  $NA_i$   $N_i^j$ ))
  (subject  $P$ )
)
```

4.3.2 Subgroups

Subgroups are created using ID certificates whose *subject* field is also a name. This can be useful in order to establish group hierarchies by means of ID certificates. However, it is worth noting that a significant difference exists between generation of subgroups and creation of ID certificates for public keys. Generation of ID certificates is normally requested by the principals involved, but subgroup certificates cannot be requested by the subgroup itself. Authorized requestors are policy-dependent, but some appropriate candidates are the naming authority defining the subgroup, or even a subgroup member. In this example, the authorized requestor is the naming authority, but this has delegated the authorization to principal R in order to avoid signing certification requests with the same private key used to generate ID certificates.

This is the request sent by R to NA_i in order to define N_l^k as subgroup of N_i^j (it is signed using the private key of R):

```
(sequence
  (cert-request
    (issuer (name  $NA_i$   $N_i^j$ ))
    (subject (name  $NA_l$   $N_l^k$ )))
  (signature ..)
)
```

Next ACL specifies that NA_l can request an ID certificate for N_i^j , and can also delegate that permission.

```
(acl
  (entry
    (subject  $NA_l$ )
    (propagate)
    (tag cert-request
      (issuer (name  $NA_i$   $N_i^j$ ))
      (subject (name  $NA_l$   $N_l^k$ )))
  )
```

```
))
)
)
```

R also sends the next authorization certificate in order to demonstrate that NA_l delegated the permission to R :

```
(cert
  (issuer  $NA_l$ )
  (subject  $R$ )
  (tag (cert-request *))
)
```

Finally, NA_i uses the data obtained from the authorization decision in order to create the certificate.

```
(cert
  (issuer (name  $NA_i$   $N_i^j$ ))
  (subject (name  $NA_l$   $N_l^k$ ))
)
```

5 Authorization Management System (AMS)

Section 2 shown a scenario where authorization certificates can be used in order to gain physical access to buildings. The system was based on delegation, and users obtained this type of certificates from trusted authorization authorities. In this section we are going to present the authorization management system, which is responsible for certification operations related to SPKI authorization and attribute certificates.

5.1 Architectural elements

NMS and AMS are based on similar architectural elements. Requestors and access points are also part of AMS. Naming authorities are replaced by authorization authorities (AA), but they share some basic functionality. AAs create attribute and authorization certificates upon the requests received through the access points or directly from the requestors.

An AMS requestor is a principal demanding the generation of a new attribute or authorization certificate. This entity must create a certification request containing information about the authorization tag (the tag is completely application-dependent). Like

in NMS, there also are two types of requestors: first, the principal requesting an authorization certificate; second, the principal requesting an attribute certificate for a particular name. As we will see later, we consider that these two situations should be managed following different approaches.

5.2 S-expressions for certification requests and ACL entries

S-expressions used in AMS to specify certification requests are also based on the structure defined by SPKI for attribute and authorization certificates. The main difference between NMS and AMS s-expressions is the *tag* field. This field contains information about the particular authorization being requested (when it is contained in a certification request) or granted (when it is part of an ACL entry).

Certification requests are also encoded as sequences composed by the request itself, and its signature.

5.3 Some examples

In order to clarify how AMS entities cooperate to generate authorization and attribute certificates, in this section we are going to analyze two certification requests. First, we explain how a principal can obtain an authorization certificate. Then, we will show how attribute certificates can be generated. In these examples, authorization policies are also represented by ACLs.

5.3.1 Authorization certificates

In this first example, *P* is a principal demanding an authorization certificate containing a tag *tag^A* from authority *AA_i*. Next certification request is created by *P*:

```
(sequence
  (cert-request
    (issuer AAi)
    (subject P)
    (tag tagA))
  (signature ..)
)
```

This request is sent to *AA_i* in order to obtain the demanded certificate. The request will be granted

if *AA_i* can find a certificate chain from its ACL entries to the requestor's public key. The authority contains the next ACL:

```
(acl
  (entry
    (subject P)
    (tag (cert-request
      (issuer AAi)
      (subject P)
      (tag tagB))
    ))
)
```

This ACL specifies that *P* can request an authorization certificate containing the permission specified by *tag^B* (*tag^A* must be more restrictive or equal to *tag^B*). Finally, the authorization authority uses the data obtained from the authorization decision in order to create the requested certificate.

```
(cert
  (issuer AAi)
  (subject P)
  (tag tagA)
)
```

One of the main advantages of this proposal is that it is possible to specify a class of certificates, possibly infinite in size, without having to issue them all. The appropriate finite subset of that class can be issued on demand. The potential infinite size of the class comes from use of **-forms*.

5.3.2 Attribute certificates

Attribute certificates can be used to specify roles. The subject can be a name defining a role, and this type of certificate states the permission related to that role. Roles can be seen as various job functions in an organization, and users can be assigned to one role depending on their responsibilities. The role permissions use to be stable since roles activities do not change frequently. However, we must answer the question: "Who must the requestor of an attribute certificate be?"

Certificates are issued by authorization authorities, hence valid requestors are those specified by their authorization policies. AMS should keep inherent

policies to a minimum, in order to allow users of the system to design their own authorization policies. Therefore, valid requestors can range from role members to specific role managers. Nevertheless, we find the latter approach very interesting for complex systems since role management can be greatly simplified using specific administrators (role managers). Authorities can authorize role managers to request attribute certificates for a particular set of group names. This authorization can be expressed as:

$$AA_i \Rightarrow RM_i^1(N_l^k, N_f^g), RM_i^n(N_j^h)$$

This expression denotes that authority AA_i authorizes role manager RM^1 to request attribute certificates for the group N^k defined by NA_l , and for the group N^g defined by NA_f . AA_i also authorizes RM^n to request this type of certificates for the group N^h defined by NA_j .

We are going to see how this relation can be implemented using AMS. In this example, RM_i^1 requests an attribute certificate for N_f^g , with the authorization tag tag^A . This is the request sent by RM_i^1 to AA_i (it is signed using the private key of RM_i^1):

```
(sequence
  (cert-request
    (issuer AA_i)
    (subject (name NA_f N_f^g))
    (tag tag^A))
  (signature ..)
)
```

The authority contains an ACL implementing the above-expressed relation. This is the ACL:

```
(acl
  (entry
    (subject RM_i^1)
    (tag (cert-request
      (issuer AA_i)
      (subject (* set
        (name NA_l N_l^k)
        (name NA_f N_f^g))))
      (tag tag^B)))
  )
  (entry
    (subject RM_i^n)
    (tag (cert-request
```

```
(issuer AA_i)
(subject (name NA_j N_j^h))
(tag tag^C)))
)
)
```

Finally, the authorization authority uses the data obtained from the authorization decision in order to create the requested certificate.

```
(cert
  (issuer AA_i)
  (subject (name NA_f N_f^g))
  (tag tag^A)
)
```

6 Use of AMBAR in DCMS

Requests and certificates are exchanged using AMBAR connections. Although other protocols like SSL (Secure Socket Layer) can be used for this purpose, we find AMBAR a valuable approach since it has been designed to exchange authorization-related information. Entities making use of AMBAR do not pay attention to issues such as the encapsulation of requests or certificates. They create AMBAR connections in order to exchange this type of information, and AMBAR modules are responsible for encapsulation and protection. Furthermore, this protocol has been designed to be session-oriented in order to optimize those scenarios where the request/response messages are exchanged between the same client and server (such as for instance, access points and authorities).

In DCMS, there are two types entities which must make use of AMBAR: access points and authorities. Requestors can request their certificates using access points, and therefore AMBAR functionality is not a requirement for them. Authorities should not employ their private keys to establish AMBAR connections since it is not suitable to protect their communications making use of the same private key signing the certificates. Authorities should generate a new key pair for communication purposes, and they should issue a certificate authorizing the new key pair to act as their *network interface*. This certificate should include a tag (`tag dcms-com`), and will be used by access points and requestors to validate that they are indeed exchanging information with the right authority.

AMBAR connections used in DCMS can perform authentication based on X.509 certificates, or SPKI certificates. Access points and authorities are always authenticated, but identity of requestors can be preserved using the anonymous mode. Credentials (additional certificates attached to the request) can be provided by access points or requestors (push method), or can be recovered from public repositories by authorities (pull method). Figure 4 shows an exchange (push) between an access point and an authorization authority, and how data are encapsulated in AMBAR messages. If the certification request is granted, the authority sends a *Resource* message containing the certificate. Otherwise, a *Negative Notification* message is generated. Negotiation is performed only once. Then, requests and results are exchanged using the previously-established channel.

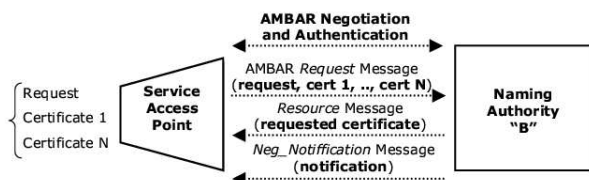


Figure 4: Communication between an access point and an authority

7 Conclusions

In this paper, we have proposed a management system that can be used in SPKI scenarios based on delegation. We present how certification requests for ID, attribute, and authorization certificates can be expressed, how authorization policies can be enforced in a distributed way, and which are the entities involved in a certification scenario.

We consider that our system provides strong mechanisms to address scalability-related problems. First, we have tried to keep inherent policies to a minimum, in order to allow system administrators to design their own authorization policies. What is more, following our approach, it is possible to specify a set of certificates without having to issue them all since they are issued on demand. Added to this, we make a clear distinction between requestors and subjects of certificates. We do not force both entities to be the same one, enabling therefore the participation of relying parties.

In order to complete our proposal, additional mechanisms must be designed, such as certificate revo-

cation or certificate storage. Currently, we are also developing a new service of DCMS for automatic reduction of certification chains. Certificate reduction can be used to improve performance of authorization decisions and, as is commented in [1], to provide anonymity services.

8 Acknowledgements

This work is partially supported by TIC2000-0198-P4-04 project (ISAIAS), and by IST-2001-32161 project (Euro6ix)

References

- [1] T. Aura and C. Ellison. Privacy and Accountability in Certificate Systems. Technical Report HUT-TCS-A61, Helsinki University of Technology, 2000.
- [2] O. Canovas and A.F. Gomez. AMBAR: Access Management Based on Authorization Reduction. In *Proceedings of the International Conference on Information and Communications security (ICICS 2001)*, volume 2229 of *Lecture Notes in Computer Science*, pages 376–380. Springer Verlag, November 2001.
- [3] O. Canovas, A.F. Gomez, H. Martinez, and G. Martinez. A Role-Based Implementation of Physical Access Control using Authorization Certificates. Technical Report UM-DITEC-2002-2, Department of Computer Engineering, University of Murcia, January 2002.
- [4] Dwaine Clarke. SPKI/SDSI HTTP Server and Certificate Chain Discovery in SPKI/SDSI . Master's thesis, M.I.T., September 2001.
- [5] Intel Corporation. *Common Data Security Architecture (CDSA)*. World Wide Web, <http://developer.intel.com/ial/security>, 2001.
- [6] J.E. Elien. Certificate discovery using SPKI/SDSI 2.0 certificates. Master's thesis, Massachusetts Institute of Technology, May 1998.
- [7] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *Simple Public Key Certificate*. IETF Internet Draft, draft-ietf-spki-cert-structure-06.txt edition, July 1999.
- [8] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *SPKI certificate*

theory, September 1999. Request For Comments (RFC) 2693.

- [9] T. Hasu and Y. Kortesniemi. *Implementing an SPKI Certificate Repository within the DNS*, Poster Paper Collection of the Theory and Practice in Public Key Cryptography (PKC 200) edition, January 2000.
- [10] J. Koponen, P. Nikander, J. Rasanen, and J. Paajarvi. Internet access through WLAN with XML encoded SPKI certificates. In *Proceedings of NordSec'00*, October 2000.
- [11] Y. Kortesniemi, T. Hasu, and J. Sars. A Revocation, Validation and Authentication Protocol for SPKI Based Delegation Systems. In *Proceedings of Network and Distributed System Security Symposium (NDSS 2000)*, February 2000.
- [12] T. Lampinen. Using SPKI Certificates for Authorization in CORBA based Distributed Object-Oriented Systems. In *Proceedings of NordSec'99*, pages 61–81, November 1999.
- [13] Per Harald Myrvang. *An Infrastructure for Authentication, Authorization and Delegation*. PhD thesis, Department of Computer Science, University of Tromso, May 2000.
- [14] R. Rivest and B. Lampson. *SDSI: A simple distributed security infrastructure*.
- [15] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2), February 1996.
- [16] K. Seamons, M. Winslett, and T. Yu. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. In *Proceedings of Network and Distributed System Security Symposium*, April 2001.