The X.509 Privilege Management Infrastructure

David CHADWICK
Information Systems Institute, University of Salford, Salford M5 4WT, England

Abstract. This paper provides an overview of the Privilege Management Infrastructure (PMI) introduced in the 2000 edition of X.509. It describes the entities in the infrastructure: Sources of Authority, Attribute Authorities and Privilege Holders, as well as the basic data structure - the attribute certificate - that is used to hold privileges. The contents of attribute certificates are described in detail, including the various policy related extensions that may be added to them. The similarities between PMIs and PKIs are highlighted. The paper also describes how attribute certificates can be used to implement the three well known access control schemes: DAC, MAC and RBAC. Finally the paper gives an overview of how a privilege verifier might operate, and the various types of information that need to be provided to it.

1. Introduction

Although attribute certificates (ACs) were first defined in X.509(97), it was not until the 4th edition of X.509 (ISO 9594-8:2001) [1] that a full infrastructure for the use of attribute certificates was defined. This infrastructure is termed a Privilege Management Infrastructure (PMI), and it enables privileges to be allocated, delegated, revoked and withdrawn in an electronic way. A PMI is to authorisation what a Public Key Infrastructure (PKI) is to authentication.

In order to understand the purpose of an X.509 PMI, it may help if we first consider how privileges are allocated and used today in a paper based privilege management system. A resource owner (e.g. the Financial Director of a company, or a General Manager), called the Source of Authority (SOA) in X.509, will sign a paper form to say that a particular person (the privilege holder) is to be allowed to use a particular resource in a particular way. For example, the Financial Director may say that a Head of Department can sign orders up to the value of ten thousand Euros, or the General Manager may sign a form authorising a user to have access to a particular restricted area of the organisation.

Paper based systems may also support delegation of authority, whereby a privilege holder is allowed to delegate the use of resources to which he has been granted access, to one or more other people. In X.509, such a privilege holder is now termed an Attribute Authority (AA), since the holder has been authorised to assign privilege attributes to other people. For example, a Head of Department may authorise a project manager to sign orders for his project up to a pre-determined sum. The Head of Department is now acting as an Attribute Authority as well as being a privilege holder himself. When performing delegation of authority, the delagator (AA) must not

overstep his authority and give the delegatee more privileges than he holds himself. For example, the Head of Departments should not give the project manager authority to spend up to a hundred thousand euros on a project, when the Head of Department is only authorised to spend up to ten thousand euros.

When a privilege is asserted (or exercised) by a privilege holder, someone (the privilege verifier) will need to check that the privilege really does belong to the person claiming the privilege. For example, if a project manager completes an Internal Requisition form in order to purchase a piece of equipment costing five thousand euros, a purchase order clerk (privilege verifier) will need to check such things as:

i) is this user allowed to sign Internal Requisitions i.e. has he been authorised to do this
ii) who gave permission for this user to sign Internal Requisitions i.e. what is the delegation chain of authority leading to this user from the Financial Directory (the Source of Authority)
iii) is this Internal Requisition within the limits of what the user is allowed to purchase i.e. did every delegator act within his limits of authority, and has the user done the same
iv) are all the signatures valid, both on the original authorisation forms and on this Internal Requisition. This is checking the authenticity of the authorisation chain, and of the Internal Requisition.
v) has the user's privilege been withdrawn. The privilege verifier will need to check that the authorisation is still current and valid, and has not been withdrawn since the original authorisation forms were signed.

The X.509 Privilege Management Infrastructure is very similar in concept to the existing paper based privilege management infrastructure, and there is direct correspondence between the entities involved.

In X.509 the Source of Authority is the resource owner and is responsible for assigning privileges to other entities. These other entities are Attribute Authorities (if they can delegate their privileges further) and Privilege Holders (if they are not allowed to delegate their privileges to anyone else). An AA is allowed to delegate privileges to other entities (both AAs and privilege holders) but may or may not be able to assert the privileges itself.

The privilege verifier is the entity that checks the asserted privileges and makes a yes/no decision as to whether the privilege may be used or not. The privilege verifier trusts the source of authority (SOA) and checks that the privilege holder has been directly or indirectly authorised by the SOA.

2. Similarities between PKIs and PMIs

Since both PMIs and PKIs are part of the same X.509 standard, one would expect some similarities to exist between the two infrastructures. In fact, there is a significant amount of overlap between the concepts and data structures used by PKIs and PMIs. These are shown in Table 1 below.

Attribute certificates (ACs) are used to hold authorisation information (privileges) in much the same way as public key certificates (PKCs) hold authentication information. The main difference is that a PKC binds a public key to its holder, whilst an AC binds a set of attributes to its holder. Only a CA can issue a

public key certificate (being a CA is a very specialised function and there are usually very few of them in an organisation), but anyone holding privileges may be an AA and be allowed to issue attribute certificates to others. This is needed in order to delegate privileges to people and is not such a specialised function as being a CA, so there might be very many AAs in an organisation.

Concept	PKI entity	PMI entity
Certificate	Public Key Certificate (PKC)	Attribute Certificate (AC)
Certificate issuer	Certification Authority (CA)	Attribute Authority (AA)
Certificate user	Subject	Holder
Certificate binding	Subject's Name to Public Key	Holder's Name to Privilege Attribute(s)
Revocation	Certificate Revocation List (CRL)	Attribute Certificate Revocation List (ACRL)
Root of trust	Root CA or Trust Anchor	Source of Authority (SOA)
Subordinate authority	Subordinate Certification Authority	Attribute Authority (AA)

Table 1. A Comparison of PKIs and PMIs

Examples of Sources of Authority might typically be the Financial Director of a company (for allocating financial privileges), or the Network Manager (for allocating networking privileges) etc. An attribute authority could then be a departmental manager, or a project manager etc.

The revocation of certificates is just as important in PMIs as in PKIs, and the certificate revocation list construct is identical for both PKIs and PMIs. It essentially holds the serial numbers of the revoked certificates, and the time of revocation.

3. The Contents of an Attribute Certificate

An attribute certificate comprises a digitally signed SEQUENCE of:

- the version number of this AC (v1 or v2)
- identification of the holder of this AC
- identification of the AA issuing this AC
- the identifier of the algorithm used to sign this AC
- the unique serial number of this AC
- the validity period of this AC
- the sequence of attributes being bound to the holder
- any optional extensions

The holder and issuer can be identified in one of three ways. Firstly by their name (termed a General Name in X.509); secondly by reference to their public key certificate; and thirdly by a hash value of a software object related to them. The General Name of a holder or issuer can be one of:

- otherName – any name of any form (this is catch all to cater for names that are not standardized in the following list)
- rfc822Name – an e-mail address as per RFC 822
- DNSName – an Internet domain name as per RFC 1035

– X.400Address – an O/R address as per X.411
– Directory Name – a directory distinguished name as per X.501
– EDIPartyName – a format agreed between EDI partners, consisting of the name of the EDI naming authority and the name of the EDI party
– UniformResourceIdentifier – a URI as per RFC 1630
– IPAddress – an Internet Protocol address as per RFC 791
– RegisteredID – any OID registered as per X.660|ISO 9834-1

If one wishes to protect the name of the AC issuer and/or holder from being present in the AC, then they can be identified by reference to their public key certificate(s). This form of identification comprises the General Name of the CA issuing the PKC, and the serial number of the PKC that was issued to them. The third way of identifying the holder and/or issuer is by a hash value of a software object related to them. The hash value (message digest) is created from an information object that can be used to directly authenticate the holder or issuer of the attribute certificate. This might be a hash of a public key belonging to the holder or issuer, or of a public key certificate issued to either of them, or of the object itself, for example if the holder is a downloadable program then the hash could be of the program itself, and the AC would then hold the privileges that should be assigned to the program. By rechecking the hash of the downloaded program one can be assured that one has got a copy of the original (virus free) software as intended by the attribute authority. When hash values are used to identify holders or issuers, then the attribute certificate says what software object the hash has been made from, and what hashing algorithm has been used to create it.

Finally, attribute certificates can have any number of extensions added to them. A standard set of extensions are defined in X.509, but other organisations can also specify their own private extensions (as is also the case with X.509 PKCs).

4. X.509 AC Extensions

Extensions to X.509 ACs are concerned with specifying some aspects of policy that govern the use and applicability of the ACs. The extensions can be split up into 5 categories:

- basic extensions
- privilege revocation extensions
- an extension to support roles
- source of authority extensions, and
- delegation of authority extensions

4.1. Basic extensions

There are 4 basic extensions that can be added to X.509 ACs. These are designed to control the time, targets, policies and users of the AC.

The time extension allows the time when the privilege can be exercised to be constrained (e.g. to certain days of the week and certain times of the day etc.)

The targeting extension allows the issuer to control the target resources that the privileges are valid for. Targets may be identified either specifically by naming

them using the General Names construct, or generically as a group of servers or services (but how group membership is defined is outside the scope of the standard).

The acceptable privilege policies extension states that the privilege can only be used with these policies that must be known to the privilege verifier associated with the target resource. This extension is always flagged as critical, meaning that if the privilege verifier does not know the policies referred to in this AC extension (or does not even recognise this extension), then it must reject the AC and deny the user access.

The user notice extension is intended to hold a message that is displayed to the holder and/or verifier when the privilege is being used i.e. displayed to the user when he/she is asserting the privilege, and displayed to the privilege verifier when he/she is validating the privilege. A similar extension is defined for public key certificates, and some well known CAs place user notices in the PKCs they issue.

4.2. Revocation Extensions

There are two extensions in this category: the CRL distribution points extension and the 'no revocation' extension.

The CRL distribution points extension points to where the revocation list or lists for this attribute certificate will be found. This allows sets of attribute certificates to have their revocation information posted to different revocation lists. This ensures that revocation lists never become too large. There are two ways of distributing CRL information. The first is by certificate serial number, in which blocks of revoked certificates (say from serial numbers 1 to 500, 501 to 1000 etc.) are posted to different CRLs and each AC in a block will contain the same distribution point extension. The second is by revocation reason, where revoked certificates are posted to different CRLs according to their reason for revocation e.g. compromised certificates list vs. privileges withdrawn list, and all ACs will hold the same set of distribution points.

The 'no revocation' extension tells the privilege verifier that this attribute certificate will never appear on an attribute certificate revocation list. This extension may be useful for short-lived attribute certificates that will never be revoked.

4.3. Extension to support Roles

If an AC is a role assignment attribute certificate (see later), a privilege verifier needs to be able to know what privileges are associated with the role in this AC. If role specification ACs are being used to specify these privileges, then the roleSpecCertIdentifier extension enables the privilege verifier to locate the role specification AC that contains the specific privileges assigned to the role in this role assignment AC. Note that this extension may not be necessary, as privilege verifiers may have their own way of locating role specification ACs, or may not even be using role specification ACs because the role specification information is held in an authorisation policy (as for example in PERMIS).

The roleSpecCertIdentifier extension contains the following fields:

- the role name (this must match the General Name of the holder in the Role Specification AC)
- the role certificate issuer (this must match the General Name of the AA signing the Role Specification AC)

- serial number of the Role Specification AC (optional)
- role certificate locator (a General Name, used to help locate the Role Specification AC)

Note that if the optional serial number is held in this extension, then if the definition of the role specification changes and a new role specification AC is issued, then all role holders will need to be issued with new role assignment ACs (since this extension will no longer be valid). So in general it is not a good idea to include the serial number in this extension.

4.4. Source of Authority Extensions

Privilege verifiers need to know who are the sources of authority for allocating privileges to the resource(s) they manage. This can be achieved in one of two ways. The first method is by out of band means. The privilege verifier in this case needs to be configured (by some trusted means) with the name (and possibly the public key) of the SOA(s) it trusts. This is similar to configuring PKI relying parties with root CAs and trust anchors.

Alternatively, it is possible to leverage the PKI to identify the trusted SOAs on behalf of the privilege verifiers. The public key certificate issued to an SOA can contain the SOA identifier extension (which does not contain any parameters). The presence of this extension in a PKC means that the CA is stating that this subject is an SOA. Thus if the privilege verifier trusts the CA to allocate sources of authority then it can reliably know who is an SOA. The privilege verifier then only needs to be configured by out of band means with the public key of the CA that it trusts (its trust anchor, or root CA).

Privilege verifiers also need to know what privilege attributes will be issued by an SOA, and if these privileges are delegated what rules govern their delegation. This information can be configured into the privilege verifier by some trusted out of band means. Alternatively, the SAO can issue a self signed “attribute descriptor” certificate, that says which attributes it will assign, and what is meant by a delegated privilege being less than a held privilege. An attribute descriptor certificate does not contain any attributes, but instead contains the attribute descriptor extension.

4.5.Delegation of Authority Extensions

There are four extensions that may be inserted into ACs to support delegation of authority.

- The basic attribute constraints extension holds a boolean to control delegation, and if set to TRUE, may contain an integer controlling further delegation
- The delegated name constraints extension restricts the names of subsequent holders of delegated privileges
- The acceptable PK certificate policies extension lists the CA policies under which subsequent delegated holders must have been authenticated
- The AA identifier extension is a back pointer to the attribute certificate of the delegating AA, to help the verifier construct a valid certificate chain

The basic attribute constraints extension comprises a boolean to say if the holder is allowed to delegate the privileges given to him/her. If the boolean is TRUE (i.e. the

holder is an AA and delegation is allowed) then it may be accompanied by an integer to say how much further delegation is allowed. A value of zero means that the AA can only issue privilege holder certificates and not AA certificates (no further delegation is allowed after this one), a value of 1 means that the holder of a delegated privilege may also be a AA and delegate privileges one more time. Larger integers correspondingly increase the number of delegations that can take place.

The delegated name constraints extension allows the AA to restrict the names of holders to whom the privilege can be delegated. The constraint consists of either "permitted subtrees" or "excluded subtrees", i.e. the holder name in assigned privileges must be within the permitted namespace or must not be within the excluded namespace.

The acceptable public key certificate policies extension is used to ensure that the holder of delegated privileges has been issued with a public key certificate under an acceptable certification policy. This is to ensure that the delegated holder has been authenticated by a PKI to a required level of trust. This helps to ensure that the PMI is not compromised by weak CAs operating lax authentication policies.

The authority attribute identifier extension is a back pointer to the attribute certificate held by the AA of this attribute certificate. This allows the privilege verifier to check that the privileges held by the AA were sufficient to allow it to assign the privileges in this delegated attribute certificate. By following the entire chain of back pointers, the verifier should eventually end up with an AC issued by a trusted SOA.

5. Applying X.509 PMIs to DAC, MAC and RBAC

5.1. Discretionary Access Controls

Various access control schemes have been devised in the past. The traditional, most popular and well-known model is the Discretionary Access Control (DAC) scheme [2]. This is typically implemented in filestores, file serves, databases etc. In the DAC scheme, users are optionally given access rights to resources by the resource administrator. In traditional systems the access rights are typically held as access control lists within each target resource. This makes it fast to compute the access control decisions. The scheme is also easy to understand, but has the disadvantage that when there are large numbers of users it can be complex to manage. Users have to be given access rights (privileges) to each resource they want to access, and when they leave or change jobs, they have to have their permissions removed from each resource (which isn't always done!).

In an X.509 PMI, the access rights may be held within the privilege attributes of attribute certificates issued to users. Each privilege attribute within an AC will describe one or more of the user's access rights. A target resource will then read a user's AC to see if he is allowed to perform the action that he is requesting.

5.2. Mandatory Access Controls

Another authorization scheme, popular with the military, is the Multilevel Secure (MLS) system, which is a type of Mandatory Access Control (MAC) scheme. In the MLS scheme, every target is given a security label, which includes a classification, and every subject is given a clearance, which includes a classification list. The

classification list specifies which type of classified target the subject is allowed to access. A typical hierarchical classification scheme used by the military is: unmarked, unclassified, restricted, confidential, secret, and top secret. A typical security policy, designed to stop information leakage, is “read down and write up”. This specifies that a subject can read targets with a lower classification than his clearance, and can write to targets with a higher classification. A user with clearance of confidential, who logs in as such, under this policy could read from unmarked to confidential targets, and write to confidential to top secret targets. The same user could also log in with a lower clearance level, say, unclassified and write to an unclassified target.

The X.509 PMI can support MLS, by allowing subjects to be given a clearance AC. The privilege attribute in the AC now holds the user’s clearance. Targets can be securely configured with their own security label and the security policy that is to direct them.

5.3. Role Based Access Controls

More recently, research has focused on Role Based Access Controls (RBAC) [3]. Role based privilege management can simplify the allocation and removal of privileges. In the basic RBAC model, a number of roles are defined. These roles typically represent organizational roles such as secretary, manager, employee etc. In the authorization policy, each role is given a set of permissions i.e. the ability to perform certain actions on certain targets. Each user is then assigned to one or more roles. When accessing a target, a user presents his role(s), and the target reads the policy to see if this role is allowed to perform this action. If the role holder changes, the new person (new role holder) inherits the same privileges as the previous role holder. By removing the role from the original role holder, his privileges are automatically removed. Many people can hold the same role, and have the same set of privileges e.g. all employees may have a basic set of privileges that are not shared by contract members of staff. Project team members may get special privileges associated with their project. When a person leaves an organisation it is usually a simpler task to remove their roles than to remove all their privileges in all the resources given to them by DAC.

The hierarchical RBAC model is a more sophisticated version of the basic RBAC model. With this model, the roles are organized hierarchically, and the senior roles inherit the privileges of the more junior roles. So for example, we might have the following hierarchy:

employee > programmer > manager > director.

If a privilege is given to an employee role e.g. can enter main building, then each of the superior roles can also enter the main building even though their role specification does not explicitly state this. If a programmer is given permission to enter the computer building, then managers and directors would also inherit this permission as well. Hierarchical roles mean that role specifications are more compact.

Another extension to basic RBAC is constrained RBAC. This allows various constraints to be applied to the role and permission assignments. One common constraint is that certain roles are declared to be mutually exclusive, meaning that the same person cannot simultaneously hold more than one role from the mutually exclusive set. For example, the roles of student and examiner, or the roles of tenderer

(one who submits a tender) and tender officer (one who opens submitted tenders) would both be examples of mutually exclusive sets. Another constraint might be placed on the number of roles a person can hold, or the number of people who can hold a particular role.

RBAC is supported by the X.509 PMI. X.509 defines role specification attribute certificates that hold the permissions granted to each role, and role assignment attribute certificates that assign various roles to the users. In role assignment ACs, the AC holder is the user, and the privilege attributes are the roles assigned to the user. Further, role assignment may be delegated if wanted.

Role specification ACs on the other hand are a special type of attribute certificate, that define the privileges that go with a role. The holder of the certificate is a role name rather than the name of a person, and the privilege attributes are the permissions granted to the role. Further, role specification certificates cannot be delegated (as can normal attribute certificates). The X.509 PMI also supports hierarchical RBAC by allowing both roles and privileges to be inserted as the privilege attributes in role specification ACs, so that the specified role inherits the privileges of the encapsulated roles.

The X.509 PMI only has a limited number of ways of supporting constrained RBAC. Time constraints can be placed on the validity period of a role assignment attribute certificate, as described earlier. Constraints can be placed on the targets at which a permission can be used, and on the policies under which an attribute certificate can confer privileges (also described earlier). Constraints can also be placed on the delegation of roles. However many of the constraints, such as the mutual exclusivity of roles, have to be enforced by mechanisms outside the standardised PMI e.g. within the privilege management policy enforcement function.

X.509 allows roles to be given to people in one of two ways. You can either put a newly standardised *role attribute* in the subjectDirectoryAttributes extension of a holder's public key certificate, or you can give the person a role assignment attribute certificate, where the attribute is the *role attribute*. Typically you would only use the former if the role was as long lived as the holder's public key certificate, otherwise you would need to re-issue their public key certificate when the person's role changes.

The management of the privileges assigned to a role, and the assignment of roles can be administered separately by different AAs if this is needed by an organisation. If this is the case, the *role attribute* can indicate which AA is responsible for allocating privileges to the role.

The newly standardised *role attribute* consists of a sequence of two parameters:

- the role authority (optional), that specifies the name of the AA that allocates the privileges to the role. This parameter will only be present if the role authority is different to the AA assigning roles to users,
- the role name, that specifies the name of the role. The values that this parameter can take are decided by the organisation using the PMI.

6. Verifying Privileges

The X.509 privilege control model is shown in Figure 1, and it comprises the following entities:

– the Privilege Asserter (AC holder) invokes some operation/service request (or object method) on a resource (the object being controlled). E.g. issues an electronic order
– the Privilege Verifier makes a yes/no decision as to whether this privilege holder can access this object in this way, based upon
  - the Privilege Policy that states precisely when the privilege verifier should conclude that a presented set of privileges is "sufficient" in order that it may grant the requested access (to the requested object, resource, application, etc.) to the privilege asserter,
  - environmental variables, if relevant, that capture other aspects of policy that are configured into the privilege verifier by some private means e.g. the current time of day, current account balances of customers etc.
  - the privileges (attribute certificates) held by the privilege asserter and/or a directory service/repository,
  - the revocation lists held in a directory service or other repository.

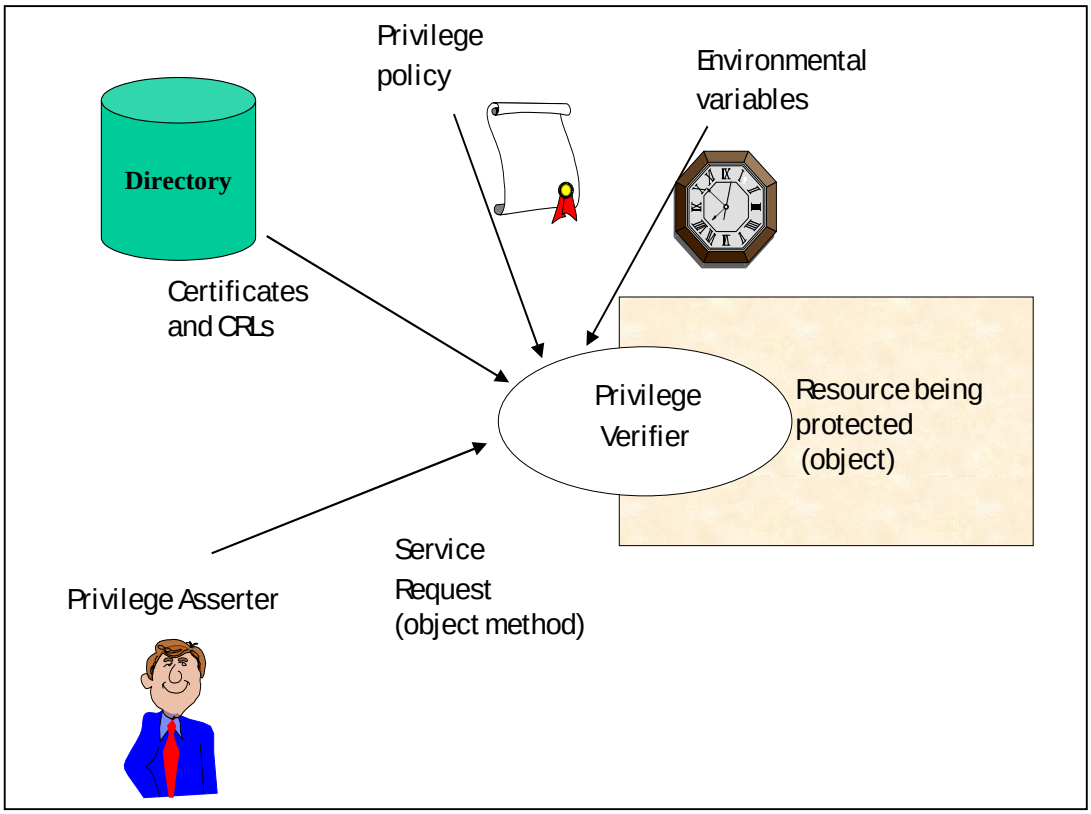

Figure 1. The Privilege Control Model

The privilege verifier needs to have access to the following information before it can verify any claimed privileges:

– the public key of the trusted root CA (this has to be configured into the verifier by some trusted means) so that it can verify signatures on the ACs and PKCs that it will evaluate,
– the name and public key of a trusted SOA, either configured into the verifier by some trusted means or via a public key certificate that can be validated

against the root CA's public key, so that the verifier can validate that ACs are issued directly or indirectly by this SOA.

- the policy rules that direct how the verifier can determine that the presented privileges are "sufficient" to access the resource, and how to determine that delegated privileges are less than held privileges (these have to be configured in by some trusted means).
- any local variables used in verifying the claimed privilege e.g. time of day. Again these have to be configured in by some trusted means.
- the attribute certificates of the privilege holder, plus a valid chain back to the SOA, plus the latest revocation information. This information may be obtained from the holder and/or a public directory service. Since the data is digitally signed it cannot be tampered with without detection and therefore does not need to be configured in by trusted means.

One can see that privilege verification is a complex process and involves at least the following steps:

- validating the AC chain back to the SOA, i.e. determining that no AC in the chain has been revoked, that each AA in the chain was allowed to delegate privileges and that the privileges were properly delegated,
- validating all the public keys used to verify the AC chain and the holder's signed request to access the resource. This includes retrieval of all the latest public key CRLs,
- checking that the claimed privilege is valid for this resource at this point in time, by checking any time restrictions, service restrictions and the policy in force,
- checking the privilege against the policy to see if it is sufficient for the mode of access being requested,
- checking the privilege against any local variables such as time of day, or credit balances to see that these are not being exceeded, and finally
- checking that the privilege holder did actually want to exercise the claimed privilege. How this is done is not specified in the standard, but one suggestion is that the request from the privilege holder to the privilege verifier could be digitally signed, and contain the ACs that the holder is wishing to use (or a pointer to them e.g. the AA name and serial number(s) of the AC(s)).

In conclusion, one can see that building a privilege management infrastructure is not a trivial task, but that once it has been built and is in use, we have a very powerful infrastructure for strong authorisation, comparable to that of PKI for authentication.

The EC PERMIS project [4] was one of the first projects to build a functioning X.509 PMI, and information about this can be found in [5][6][7].

7. References

[1] ITU-T Rec. X.509 (2000) | ISO/IEC 9594-8 The Directory: Authentication Framework

[2] Sandhu, R. and Samarati, P. "Access controls, principles and practice". IEEE Communications, 32(9), pp 40-48, 1994

[3] Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E. "Role Based Access Control Models". IEEE Computer 29, 2 (Feb 1996), p38-43.

[4] The EC PERMIS, see http://www.permis.org and http://sec.isi.salford.ac.uk/permis

[5] D.W.Chadwick, A. Otenko, E.Ball. “Implementing Role Based Access Controls Using X.509 Attribute Certificates”, IEEE Internet Computing, March-April 2003, pp. 62-69.

[6] D.W.Chadwick, A. Otenko “The PERMIS X.509 Role Based Privilege Management Infrastructure”. Future Generation Computer Systems, 936 (2002) 1–13, December 2002. Elsevier Science BV.

[7] D.W.Chadwick, A. Otenko. “RBAC Policies in XML for X.509 Based Privilege Management” in Security in the Information Society: Visions and Perspectives: IFIP TC11 17th Int. Conf. On Information Security (SEC2002), May 7-9, 2002, Cairo, Egypt. Ed. by M. A. Ghonaimy, M. T. El-Hadidi, H.K.Aslan, Kluwer Academic Publishers, pp 39-53.