Understanding X.509 Attribute Certificates
Pascal Jakobi, Systems Architect, Thales

Problem statement

One of the most significant trends in today's computing is the globalization of the Internet. The distinction between the Internet and corporate intranet tends to disappear over time as cloud computing and mobilization factors drive us into network expansion and fragmentation with increasing rapidity.

This globalization trend has various consequences – most of them will not be discussed in this document. The key aspect that will be covered here is its impact on access control technologies. More than ever, the need for open standards is clear.

There are many different access control techniques in the market. However, clearly, Role Based Access Control (RBAC) and derived techniques such as Attribute Based Access Control (ABAC) dominate. But the open standards problem remains, as most RBAC solutions are heavily proprietary. This will cause an interoperability nightmare, at the time when everyone in the industry tries to connect heterogeneous systems together.

And this is where X.509 attribute certificates (ACs), along with other technologies such as XACML, might help.

Standards based Access Control

About XACML

Standards for Access Control are not so many. The most widely known (and supported) access control policy language is XACML from OASIS. XACML stands for eXtended Access Control Markup Language – this is an XML dialect that will be used between requesting parties and authorization servers. There are already numerous documents that cover XACML – let the reader refer to those for details about this technology.

In the XACML language, the requesting party will rely on a Policy Enforcement Point (a library, a proxy...) that sends XACML requests towards an authorization server (a Policy Decision Point) and handles responses.

However, one may wonder how does the PDP “decide” to authorize or not a given operation.

As a matter of fact, this is outside the strict scope of XACML. However the standard describes PDPs as logical machines that validate requests against security policies (also expressed via XACML), and contain references to external attributes. An attribute may be some environmental parameter (such as the computer's load) or requester characteristics. More frequently it is

Illustration 1: XACML architecture
contains characteristics of the requester – which are often found within the corporate Directory.

**XACML vs. RBAC**

As we just saw, a PDP is defined simply as an authorization server that may be accessed with XACML requests/responses.

The PDP stores expressions that use XACML to implement rules such as “*if user has role supervisor or above, then grant him permission to access financial files*”. Our purpose is not to cover the XACML syntax for security policies here. The reader may find many documents to explain the concepts of XACML on the Internet, starting with OASIS.

What is certain however, is that at some point in the policy life cycle, we will have to express a *role* attribute:

```xml
<Attribute AttributeId="role"  
  DataType="http://www.w3.org/2001/XMLSchema#string">
  <AttributeValue>supervisor</AttributeValue>
</Attribute>
```

The processing of rules continues as follows. Based on an *attributeId*, a PDP will be able to understand that the user's attribute is to be retrieved from an RBAC subsystem. For example, the sunXACML open source PDP provides the notion of *attribute finders* that are governed by such ids.

How this role is extracted from a data source is of course not specified in XACML. X.509 is one way among others to carry roles.

This being said, the ANSI has published a standard that describes expected features from RBAC implementations and proposes an “abstract” API. Not surprisingly, that API includes an *AssignedRoles* function, that can be used to retrieve all roles that are assigned to a given user....

---

**About X.509 privileges management**

The *X.509 standard from the International Telecommunications Union* is now widely recognized within the computer industry. Essentially, this standard has been the theoretical foundation for the entire PKI/smart cards/security, etc. segment of the industry.

What is less known is that X.509's scope also covers the needs of privileges management and access control. As a matter of fact, section 3 of the standard (“*Attribute Certificate Framework*”) provides a comprehensive description of the software components needed in order to deal with attributes, i.e. pieces of information that characterize a Directory object (generally a person). This work has been complemented by some IETF RFCs, such as RFC 5755 and RFC 5280.

**Groups or certificates ?**

We have shown that privileges/role information has to be provided to XACML PDPs in order to provide a coherent RBAC service. We now need to examine what the actual data structure must look like in order to support role based definitions.

Most solutions are based on group membership. For example a given user would have the “*supervisor*” role because it would be a member of the corresponding Directory group.

But this approach is inherently flawed. First, group membership does not allow for “attributes” to be associated to the group assignment. This means that there is not a simple way to implement rich...
access control features such as delegation or time constraints (a role is allocated to a person within a given time range – say weekdays). Also, security is inherently weak because there is not an easy way to digitally sign a group membership which places strains on trust assertions that are based on these group associations.

All this explains why the need for an X.509 approach to role assertions exists today for systems that utilize XACML based access control mechanisms. The basic idea is to provide authorized users of a given role a signed document, or attribute certificate, that describes the properties that correspond to their role usage. These properties include role name, delegation attributes, temporal constraints, signature attributes, and more. X.509 is one technology that can also be used for secure transfer of authorization attributes between domains.

**X.509 attribute and public-key certificates**

The X.509 standard covers 2 types of certificate frameworks: public-key and attribute certificates. Both are logically linked to a Directory user (in the vast majority of cases).

It is important to distinguish these 2 notions. In fact, a public-key certificate (PKC) mimics a standard passport whereas an AC can be viewed as a visa to enter a country.

PKCs are used for authentication and are essentially a means for its owner to identify himself, with a reasonable level of security, by binding his name to a public key.

Attribute certificates, on the other hand, are used for authorization as they bind properties to the name of its bearer. For example, an attribute certificate could be delivered to all system administrators within a company, each bearing the name of the particular administrator. These certificates, in turn, could be required to access some dedicated web application.

The duration for usage of attribute certificates is generally different to that of PKCs - e.g. a "driving license" AC could be valid for 5 years, whilst a “conference delegate” AC could be valid for 3 days. Also their life cycle is often much more dynamic: revocation, delegation is much more frequent.

Furthermore, ACs can be delegated between users e.g. a system administrator might delegate his role to a colleague whilst he is away on holiday. PKCs on the other hand cannot be delegated by users.

Lastly, the PKC issuer (the Certification authority) is not usually authoritative for the authorization information. CAs and source of authorization are separate roles in most cases.

An example might clarify such a statement.

Imagine a person named J. Smith that owns a PKC from its MoD and an AC stating that he has the role “Manager”. His PKC then “proves” that he is who he claims to be (provided that we trust the MoD CA). On the other hand, his AC certificate only proves that “J. Smith” has a “Manager” role within the MoD.

**Attribute Certificates contents**

Both PKCs and ACs are ASN.1 encoded binary structures, and mostly, carry the following information:

- **Holder/Subject** – the identity of the user, usually an LDAP/X.500 distinguished name. For
an AC this can also be a public key.

- **Issuer** – the identity of the certificate issuer (i.e. the CA for a PKC, or AA for an AC). In most cases, this is the LDAP/X.500 distinguished name of the issuer, but it could identify the issuer by pointing to its PKC.

- **Algorithm identifier and Signature** – which digitally protect the certificate

- **Serial Number** – a unique number assigned by the issuer

- **Validity Period** - not before time, not after time.

- **Attributes (AC only)** – Typically, an AC attribute might convey either security clearances that are so frequently used in the military (“unclassified”, “secret”, etc.) or it may convey a role such as “Administrator”, etc.

- **Public key (PKC only)**

- **Extensions** – these contain policy and other information to control the use of the certificate.

**AC extensions**

Many potential extensions for certificates have been defined in X.509 (see §15 of the standard). Below is a brief description of the most frequently used extensions.

*Time specifications* provide a means to limit the use of a privilege to a particular period of time (a privilege AC would be usable only on weekdays, for example).

*Single use* provides the means to create certificates that may be used only once.

*AC targeting* may be used to logically link an attribute to a specified number of servers, services, etc. For example, targeting is usable to narrow the “systems administrator” role to a given DNS domain only.

Finally the *group AC* extension, provides the means to issue an attribute certificate to a group of entities, rather than to a single one. Using this extension will vastly simplify large-scale deployments.

The *indirectIssuer* extension means that the certificate holder is authorized to issues ACs on behalf of other AAs. The *issuedOnBehalf* extension is inserted in ACs by an indirect issuer in order to be able to construct a delegation chain.

Many other extensions have been defined, demonstrating that X.509 is extensible by design.

**Privilege Management Infrastructures**

Managing Attribute Certificates (issuance, revocation...) requires some specialized software.

The PMIs theory of operations can be summarized as follows:

- The SOA is the initial issuer of ACs and is equivalent to root CAs for PKCs. Its PKC may have an *soaIdentifier* extension.

- Some privilege holders are authorized to act as an Attribute Authorities (AAs).
These top-level AAs may of course create subordinate authorities, thus creating a *delegation path*.

- At last, AAs deliver ACs to end entities in turn. These entities may not delegate their privileges further.
- Privilege verifiers check the various ACs whenever needed. This check is based on trusting the SOA (this means that verifiers need to have the SOA PKC by one means or another). The actual verifier process is significantly complex and documented at length in §16 of X.509.

**Delegation service**

Briefly stated, delegation is a simple operation where an authorized user X grants part of his privileges to another user Y, for a possibly shorter period of time.

However, this simple scheme has a major drawback. As all ACs have to be signed, this means that all users who would like to delegate privileges (AAs) must possess PKCs. In other words, deployment of a PKI would become a prerequisite before using Attribute Certificates, as shown in the diagram below.

In order to circumvent this (major) issue, X.509 defines a delegation service that can delegate ACs on behalf of AAs without requiring the AAs to have PKCs themselves. The basic idea is to introduce a “relay” (Web) service that signs ACs and sets the *onBehalfOf* extension of an AC to the original issuer's Directory name.
In the DS architectures, only one PKC is mandated, at the service level. This leaves authentication of the issuers to the DS to be achieved using whatever is considered as sufficiently secure (and compatible with WS Security standard).

**Foreword**

The goal of this document will have been reached his objective if the reader understands the value of the X.509 architecture. However let him be conscious that products that implement these features are still infrequent.

The industry and the Open Source community have work to do in order to achieve full-scale, credible solutions in this technology space. Hopefully, some of these open technologies will emerge in the coming years. Today it can be noted that at least some projects have started around those concepts.

The author wishes to thank S. Mc Kinnney and D. Chadwick for their help.