Network Working Group
Request for Comments: 3281
Category: Standards Track

S. Farrell
Baltimore Technologies
R. Housley
RSA Laboratories

April 2002

An Internet Attribute Certificate Profile for Authorization

Status of this Memo

Copyright Notice

Abstract

This specification defines a profile for the use of X.509 Attribute Certificates in Internet Protocols. Attribute certificates may be used in a wide range of applications and environments covering a broad spectrum of interoperability goals and a broader spectrum of operational and assurance requirements. The goal of this document is to establish a common baseline for generic applications requiring broad interoperability as well as limited special purpose requirements. The profile places emphasis on attribute certificate support for Internet electronic mail, IPSec, and WWW security applications.

Table of Contents

1. Introduction

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119.

X.509 public key certificates (PKCs) [X.509-1997, X.509-2000, PKIXPROF] bind an identity and a public key. An attribute certificate (AC) is a structure similar to a PKC; the main difference being that the AC contains no public key. An AC may contain

attributes that specify group membership, role, security clearance, or other authorization information associated with the AC holder. The syntax for the AC is defined in Recommendation X.509, making the term "X.509 certificate" ambiguous.

Some people constantly confuse PKCs and ACs. An analogy may make the distinction clear. A PKC can be considered to be like a passport: it identifies the holder, tends to last for a long time, and should not be trivial to obtain. An AC is more like an entry visa: it is typically issued by a different authority and does not last for as long a time. As acquiring an entry visa typically requires presenting a passport, getting a visa can be a simpler process.

Authorization information may be placed in a PKC extension or placed in a separate attribute certificate (AC). The placement of authorization information in PKCs is usually undesirable for two reasons. First, authorization information often does not have the same lifetime as the binding of the identity and the public key. When authorization information is placed in a PKC extension, the general result is the shortening of the PKC useful lifetime. Second, the PKC issuer is not usually authoritative for the authorization information. This results in additional steps for the PKC issuer to obtain authorization information from the authoritative source.

For these reasons, it is often better to separate authorization information from the PKC. Yet, authorization information also needs to be bound to an identity. An AC provides this binding; it is simply a digitally signed (or certified) identity and set of attributes.

An AC may be used with various security services, including access control, data origin authentication, and non-repudiation.

PKCs can provide an identity to access control decision functions. However, in many contexts the identity is not the criterion that is used for access control decisions, rather the role or group-membership of the accessor is the criterion used. Such access control schemes are called role-based access control.

When making an access control decision based on an AC, an access control decision function may need to ensure that the appropriate AC holder is the entity that has requested access. One way in which the linkage between the request or identity and the AC can be achieved is the inclusion of a reference to a PKC within the AC and the use of the private key corresponding to the PKC for authentication within the access request.

ACs may also be used in the context of a data origin authentication service and a non-repudiation service. In these contexts, the attributes contained in the AC provide additional information about the signing entity. This information can be used to make sure that the entity is authorized to sign the data. This kind of checking depends either on the context in which the data is exchanged or on the data that has been digitally signed.

1.1 Delegation and AC chains

The X.509 standard [X.509-2000] defines authorization as the "conveyance of privilege from one entity that holds such privilege, to another entity". An AC is one authorization mechanism.

An ordered sequence of ACs could be used to verify the authenticity of a privilege asserter's privilege. In this way, chains or paths of ACs could be employed to delegate authorization.

Since the administration and processing associated with such AC chains is complex and the use of ACs in the Internet today is quite limited, this specification does NOT RECOMMEND the use of AC chains. Other (future) specifications may address the use of AC chains. This specification deals with the simple cases, where one authority issues all of the ACs for a particular set of attributes. However, this simplification does not preclude the use of several different authorities, each of which manages a different set of attributes. For example, group membership may be included in one AC issued by one authority, and security clearance may be included in another AC issued by another authority.

This means that conformant implementations are only REQUIRED to be able to process a single AC at a time. Processing of more than one AC, one after another, may be necessary. Note however, that validation of an AC MAY require validation of a chain of PKCs, as specified in [PKIXPROF].

1.2 Attribute Certificate Distribution ("push" vs. "pull")

As discussed above, ACs provide a mechanism to securely provide authorization information to, for example, access control decision functions. However, there are a number of possible communication paths for ACs.

In some environments, it is suitable for a client to "push" an AC to a server. This means that no new connections between the client and server are required. It also means that no search burden is imposed on servers, which improves performance and that the AC verifier is

only presented with what it "needs to know." The "push" model is especially suitable in inter-domain cases where the client's rights should be assigned within the client's "home" domain.

In other cases, it is more suitable for a client to simply authenticate to the server and for the server to request or "pull" the client's AC from an AC issuer or a repository. A major benefit of the "pull" model is that it can be implemented without changes to the client or to the client-server protocol. The "pull" model is especially suitable for inter-domain cases where the client's rights should be assigned within the server's domain, rather than within the client's domain.

There are a number of possible exchanges involving three entities: the client, the server, and the AC issuer. In addition, a directory service or other repository for AC retrieval MAY be supported.

Figure 1 shows an abstract view of the exchanges that may involve ACs. This profile does not specify a protocol for these exchanges.

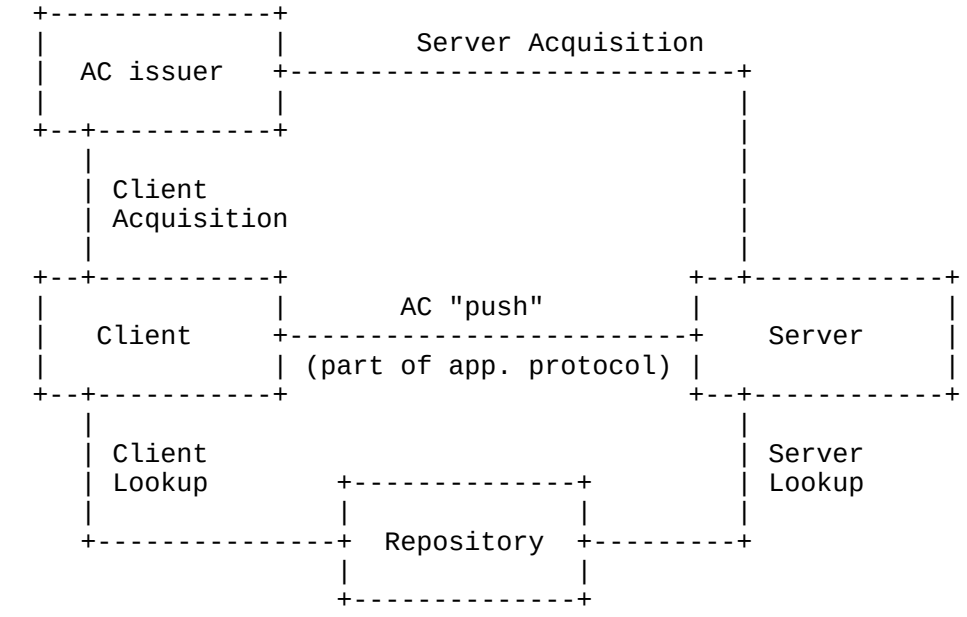

Figure 1: AC Exchanges

1.3 Document Structure

Section 2 defines some terminology. Section 3 specifies the requirements that this profile is intended to meet. Section 4 contains the profile of the X.509 AC. Section 5 specifies rules for AC validation. Section 6 specifies rules for AC revocation checks. Section 7 specifies optional features which MAY be supported; however, support for these features is not required for conformance to this profile. Finally, appendices contain the list of OIDs required to support this specification and an ASN.1 module.

2. Terminology

For simplicity, we use the terms client and server in this specification. This is not intended to indicate that ACs are only to be used in client-server environments. For example, ACs may be used in the S/MIME v3 context, where the mail user agent would be both a "client" and a "server" in the sense the terms are used here.

Term	Meaning
AA	Attribute Authority, the entity that issues the AC, synonymous in this specification with "AC issuer"
AC	Attribute Certificate
AC user	any entity that parses or processes an AC
AC verifier	any entity that checks the validity of an AC and then makes use of the result
AC issuer	the entity which signs the AC, synonymous in this specification with "AA"
AC holder	the entity indicated (perhaps indirectly) in the holder field of the AC
Client	the entity which is requesting the action for which authorization checks are to be made
Proxying	In this specification, Proxying is used to mean the situation where an application server acts as an application client on behalf of a user. Proxying here does not mean granting of authority.
PKC	Public Key Certificate - uses the type ASN.1 Certificate defined in X.509 and profiled in RFC 2459. This (non-standard) acronym is used in order to avoid confusion about the term "X.509 certificate".
Server	the entity which requires that the authorization checks are made

3. Requirements

This AC profile meets the following requirements.

Time/Validity requirements:

1. Support for short-lived as well as long-lived ACs. Typical short-lived validity periods might be measured in hours, as opposed to months for PKCs. Short validity periods allow ACs to be useful without a revocation mechanism.

Attribute Types:

2. Issuers of ACs should be able to define their own attribute types for use within closed domains.

3. Some standard attribute types, which can be contained within ACs, should be defined. Examples include "access identity," "group," "role," "clearance," "audit identity," and "charging identity."

4. Standard attribute types should be defined in a manner that permits an AC verifier to distinguish between uses of the same attribute in different domains. For example, the "Administrators group" as defined by Baltimore and the "Administrators group" as defined by SPYRUS should be easily distinguished.

Targeting of ACs:

5. It should be possible to "target" an AC at one, or a small number of, servers. This means that a trustworthy non-target server will reject the AC for authorization decisions.

Push vs. Pull

6. ACs should be defined so that they can either be "pushed" by the client to the server, or "pulled" by the server from a repository or other network service, including an online AC issuer.

4. Attribute Certificate Profile

ACs may be used in a wide range of applications and environments covering a broad spectrum of interoperability goals and a broader spectrum of operational and assurance requirements. The goal of this document is to establish a common baseline for generic applications requiring broad interoperability and limited special purpose

requirements. In particular, the emphasis will be on supporting the use of attribute certificates for informal Internet electronic mail, IPSec, and WWW applications.

This section presents a profile for ACs that will foster interoperability. This section also defines some private extensions for the Internet community.

While the ISO/IEC/ITU documents use the 1993 (or later) version of ASN.1, this document uses the 1988 ASN.1 syntax, as has been done for PKCs [PKIXPROF]. The encoded certificates and extensions from either ASN.1 version are bit-wise identical.

Where maximum lengths for fields are specified, these lengths refer to the DER encoding and do not include the ASN.1 tag or length fields.

Conforming implementations MUST support the profile specified in this section.

4.1 X.509 Attribute Certificate Definition

X.509 contains the definition of an AC given below. All types that are not defined in this document can be found in [PKIXPROF].

```
AttributeCertificate ::= SEQUENCE {
     acinfo               AttributeCertificateInfo,
     signatureAlgorithm   AlgorithmIdentifier,
     signatureValue       BIT STRING
}

AttributeCertificateInfo ::= SEQUENCE {
     version              AttCertVersion -- version is v2,
     holder               Holder,
     issuer               AttCertIssuer,
     signature            AlgorithmIdentifier,
     serialNumber         CertificateSerialNumber,
     attrCertValidityPeriod   AttCertValidityPeriod,
     attributes           SEQUENCE OF Attribute,
     issuerUniqueID       UniqueIdentifier OPTIONAL,
     extensions           Extensions OPTIONAL
}

AttCertVersion ::= INTEGER { v2(1) }
Holder ::= SEQUENCE {
      baseCertificateID   [0] IssuerSerial OPTIONAL,
               -- the issuer and serial number of
               -- the holder's Public Key Certificate
```

```
        entityName          [1] GeneralNames OPTIONAL,
                  -- the name of the claimant or role
        objectDigestInfo    [2] ObjectDigestInfo OPTIONAL
                  -- used to directly authenticate the holder,
                  -- for example, an executable
}

ObjectDigestInfo ::= SEQUENCE {
     digestedObjectType  ENUMERATED {
             publicKey            (0),
             publicKeyCert        (1),
             otherObjectTypes     (2) },
                     -- otherObjectTypes MUST NOT
                     -- be used in this profile
     otherObjectTypeID   OBJECT IDENTIFIER OPTIONAL,
     digestAlgorithm     AlgorithmIdentifier,
     objectDigest        BIT STRING
}

AttCertIssuer ::= CHOICE {
     v1Form   GeneralNames,  -- MUST NOT be used in this
                             -- profile
     v2Form   [0] V2Form     -- v2 only
}

V2Form ::= SEQUENCE {
     issuerName            GeneralNames  OPTIONAL,
     baseCertificateID     [0] IssuerSerial  OPTIONAL,
     objectDigestInfo      [1] ObjectDigestInfo  OPTIONAL
       -- issuerName MUST be present in this profile
       -- baseCertificateID and objectDigestInfo MUST NOT
       -- be present in this profile
}

IssuerSerial  ::=  SEQUENCE {
     issuer         GeneralNames,
     serial         CertificateSerialNumber,
     issuerUID      UniqueIdentifier OPTIONAL
}

AttCertValidityPeriod  ::= SEQUENCE {
     notBeforeTime  GeneralizedTime,
     notAfterTime   GeneralizedTime
}
```

Although the Attribute syntax is defined in [PKIXPROF], we repeat the definition here for convenience.

```
Attribute ::= SEQUENCE {
      type      AttributeType,
      values    SET OF AttributeValue
        -- at least one value is required
}

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY DEFINED BY AttributeType
```

Implementers should note that the DER encoding (see [X.509-1988],[X.208-1988]) of the SET OF values requires ordering of the encodings of the values. Though this issue arises with respect to distinguished names, and has to be handled by [PKIXPROF] implementations, it is much more significant in this context, since the inclusion of multiple values is much more common in ACs.

4.2 Profile of Standard Fields

GeneralName offers great flexibility. To achieve interoperability, in spite of this flexibility, this profile imposes constraints on the use of GeneralName.

Conforming implementations MUST be able to support the dNSName, directoryName, uniformResourceIdentifier, and iPAddress options. This is compatible with the GeneralName requirements in [PKIXPROF] (mainly in section 4.2.1.7).

Conforming implementations MUST NOT use the x400Address, ediPartyName, or registeredID options.

Conforming implementations MAY use the otherName option to convey name forms defined in Internet Standards. For example, Kerberos [KRB] format names can be encoded into the otherName, using a Kerberos 5 principal name OID and a SEQUENCE of the Realm and the PrincipalName.

4.2.1 Version

The version field MUST have the value of v2. That is, the version field is present in the DER encoding.

Note: This version (v2) is not backwards compatible with the previous attribute certificate definition (v1) from the 1997 X.509 standard [X.509-1997], but is compatible with the v2 definition from X.509 (2000) [X.509-2000].

4.2.2 Holder

The Holder field is a SEQUENCE allowing three different (optional) syntaxes: baseCertificateID, entityName and objectDigestInfo. Where only one option is present, the meaning of the Holder field is clear. However, where more than one option is used, there is a potential for confusion as to which option is "normative", which is a "hint" etc. Since the correct position is not clear from [X.509-2000], this specification RECOMMENDS that only one of the options be used in any given AC.

For any environment where the AC is passed in an authenticated message or session and where the authentication is based on the use of an X.509 PKC, the holder field SHOULD use the baseCertificateID.

With the baseCertificateID option, the holder's PKC serialNumber and issuer MUST be identical to the AC holder field. The PKC issuer MUST have a non-empty distinguished name which is to be present as the single value of the holder.baseCertificateID.issuer construct in the directoryName field. The AC holder.baseCertificateID.issuerUID field MUST only be used if the holder's PKC contains an issuerUniqueID field. If both the AC holder.baseCertificateID.issuerUID and the PKC issuerUniqueID fields are present, the same value MUST be present in both fields. Thus, the baseCertificateID is only usable with PKC profiles (like [PKIXPROF]) which mandate that the PKC issuer field contain a non-empty distinguished name value.

Note: An empty distinguished name is a distinguished name where the SEQUENCE OF relative distinguished names is of zero length. In a DER encoding, this has the value '3000'H.

If the holder field uses the entityName option and the underlying authentication is based on a PKC, the entityName MUST be the same as the PKC subject field or one of the values of the PKC subjectAltName field extension (if present). Note that [PKIXPROF] mandates that the subjectAltName extension be present if the PKC subject is an empty distinguished name. See the security considerations section which mentions some name collision problems that may arise when using the entityName option.

In any other case where the holder field uses the entityName option, only one name SHOULD be present.

Implementations conforming to this profile are not required to support the use of the objectDigest field. However, section 7.3 specifies how this optional feature MAY be used.

Any protocol conforming to this profile SHOULD specify which AC holder option is to be used and how this fits with the supported authentication schemes defined in that protocol.

4.2.3 Issuer

ACs conforming to this profile MUST use the v2Form choice, which MUST contain one and only one GeneralName in the issuerName, which MUST contain a non-empty distinguished name in the directoryName field. This means that all AC issuers MUST have non-empty distinguished names. ACs conforming to this profile MUST omit the baseCertificateID and objectDigestInfo fields.

Part of the reason for the use of the v2Form containing only an issuerName is that it means that the AC issuer does not have to know which PKC the AC verifier will use for it (the AC issuer). Using the baseCertificateID field to reference the AC issuer would mean that the AC verifier would have to trust the PKC that the AC issuer chose (for itself) at AC creation time.

4.2.4 Signature

Contains the algorithm identifier used to validate the AC signature.

This MUST be one of the signing algorithms defined in [PKIXALGS]. Conforming implementations MUST honor all MUST/SHOULD/MAY signing algorithm statements specified in [PKIXALGS].

4.2.5 Serial Number

For any conforming AC, the issuer/serialNumber pair MUST form a unique combination, even if ACs are very short-lived.

AC issuers MUST force the serialNumber to be a positive integer, that is, the sign bit in the DER encoding of the INTEGER value MUST be zero - this can be done by adding a leading (leftmost) ’00’H octet if necessary. This removes a potential ambiguity in mapping between a string of octets and an integer value.

Given the uniqueness and timing requirements above, serial numbers can be expected to contain long integers. AC users MUST be able to handle serialNumber values longer than 4 octets. Conformant ACs MUST NOT contain serialNumber values longer than 20 octets.

There is no requirement that the serial numbers used by any AC issuer follow any particular ordering. In particular, they need not be monotonically increasing with time. Each AC issuer MUST ensure that each AC that it issues contains a unique serial number.

4.2.6 Validity Period

The attrCertValidityPeriod (a.k.a. validity) field specifies the period for which the AC issuer certifies that the binding between the holder and the attributes fields will be valid.

The generalized time type, GeneralizedTime, is a standard ASN.1 type for variable precision representation of time. The GeneralizedTime field can optionally include a representation of the time differential between the local time zone and Greenwich Mean Time.

For the purposes of this profile, GeneralizedTime values MUST be expressed in Coordinated universal time (UTC) (also known as Greenwich Mean Time or Zulu)) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even when the number of seconds is zero. GeneralizedTime values MUST NOT include fractional seconds.

(Note: this is the same as specified in [PKIXPROF], section 4.1.2.5.2.)

AC users MUST be able to handle an AC which, at the time of processing, has parts of its validity period or all its validity period in the past or in the future (a post-dated AC). This is valid for some applications, such as backup.

4.2.7 Attributes

The attributes field gives information about the AC holder. When the AC is used for authorization, this will often contain a set of privileges.

The attributes field contains a SEQUENCE OF Attribute. Each Attribute MAY contain a SET OF values. For a given AC, each AttributeType OBJECT IDENTIFIER in the sequence MUST be unique. That is, only one instance of each attribute can occur in a single AC, but each instance can be multi-valued.

AC users MUST be able to handle multiple values for all attribute types.

An AC MUST contain at least one attribute. That is, the SEQUENCE OF Attributes MUST NOT be of zero length.

Some standard attribute types are defined in section 4.4.

4.2.8 Issuer Unique Identifier

This field MUST NOT be used unless it is also used in the AC issuer's PKC, in which case it MUST be used. Note that [PKIXPROF] states that this field SHOULD NOT be used by conforming CAs, but that applications SHOULD be able to parse PKCs containing the field.

4.2.9 Extensions

The extensions field generally gives information about the AC as opposed to information about the AC holder.

An AC that has no extensions conforms to the profile; however, section 4.3 defines the extensions that MAY be used with this profile, and whether or not they may be marked critical. If any other critical extension is used, the AC does not conform to this profile. However, if any other non-critical extension is used, the AC does conform to this profile.

The extensions defined for ACs provide methods for associating additional attributes with holders. This profile also allows communities to define private extensions to carry information unique to those communities. Each extension in an AC may be designated as critical or non-critical. An AC using system MUST reject an AC if it encounters a critical extension it does not recognize; however, a non-critical extension may be ignored if it is not recognized. Section 4.3 presents recommended extensions used within Internet ACs and standard locations for information. Communities may elect to use additional extensions; however, caution should be exercised in adopting any critical extensions in ACs which might prevent use in a general context.

4.3 Extensions

4.3.1 Audit Identity

In some circumstances, it is required (e.g. by data protection/data privacy legislation) that audit trails not contain records which directly identify individuals. This circumstance may make the use of the AC holder field unsuitable for use in audit trails.

To allow for such cases, an AC MAY contain an audit identity extension. Ideally it SHOULD be infeasible to derive the AC holder's identity from the audit identity value without the cooperation of the AC issuer.

The value of the audit identity, along with the AC issuer/serial, SHOULD then be used for audit/logging purposes. If the value of the audit identity is suitably chosen, a server/service administrator can use audit trails to track the behavior of an AC holder without being able to identify the AC holder.

The server/service administrator in combination with the AC issuer MUST be able to identify the AC holder in cases where misbehavior is detected. This means that the AC issuer MUST be able to determine the actual identity of the AC holder from the audit identity.

Of course, auditing could be based on the AC issuer/serial pair; however, this method does not allow tracking of the same AC holder with multiple ACs. Thus, an audit identity is only useful if it lasts for longer than the typical AC lifetime. Auditing could also be based on the AC holder's PKC issuer/serial; however, this will often allow the server/service administrator to identify the AC holder.

As the AC verifier might otherwise use the AC holder or some other identifying value for audit purposes, this extension MUST be critical when used.

Protocols that use ACs will often expose the identity of the AC holder in the bits on-the-wire. In such cases, an opaque audit identity does not make use of the AC anonymous; it simply ensures that the ensuing audit trails do not contain identifying information.

The value of an audit identity MUST be longer than zero octets. The value of an audit identity MUST NOT be longer than 20 octets.

name	id-pe-ac-auditIdentity
OID	{ id-pe 4 }
syntax	OCTET STRING
criticality	MUST be TRUE

4.3.2 AC Targeting

To target an AC, the target information extension, imported from [X.509-2000], MAY be used to specify a number of servers/services. The intent is that the AC SHOULD only be usable at the specified servers/services. An (honest) AC verifier who is not amongst the named servers/services MUST reject the AC.

If this extension is not present, the AC is not targeted and may be accepted by any server.

In this profile, the targeting information simply consists of a list of named targets or groups.

The following syntax is used to represent the targeting information:

```
Targets ::= SEQUENCE OF Target

Target  ::= CHOICE {
  targetName          [0] GeneralName,
  targetGroup         [1] GeneralName,
  targetCert          [2] TargetCert
}

TargetCert  ::= SEQUENCE {
  targetCertificate    IssuerSerial,
  targetName           GeneralName OPTIONAL,
  certDigestInfo       ObjectDigestInfo OPTIONAL
}
```

The targetCert CHOICE within the Target structure is only present to allow future compatibility with [X.509-2000] and MUST NOT be used.

The targets check passes if the current server (recipient) is one of the targetName fields in the Targets SEQUENCE, or if the current server is a member of one of the targetGroup fields in the Targets SEQUENCE. In this case, the current server is said to "match" the targeting extension.

How the membership of a target within a targetGroup is determined is not defined here. It is assumed that any given target "knows" the names of the targetGroups to which it belongs or can otherwise determine its membership. For example, the targetGroup specifies a DNS domain, and the AC verifier knows the DNS domain to which it belongs. For another example, the targetGroup specifies "PRINTERS," and the AC verifier knows whether or not it is a printer or print server.

Note: [X.509-2000] defines the extension syntax as a "SEQUENCE OF Targets". Conforming AC issuer implementations MUST only produce one "Targets" element. Confirming AC users MUST be able to accept a "SEQUENCE OF Targets". If more than one Targets element is found in an AC, the extension MUST be treated as if all Target elements had been found within one Targets element.

name	id-ce-targetInformation
OID	{ id-ce 55 }
syntax	SEQUENCE OF Targets
criticality	MUST be TRUE

4.3.3 Authority Key Identifier

The authorityKeyIdentifier extension, as profiled in [PKIXPROF], MAY be used to assist the AC verifier in checking the signature of the AC. The [PKIXPROF] description should be read as if "CA" meant "AC issuer." As with PKCs, this extension SHOULD be included in ACs.

Note: An AC, where the issuer field used the baseCertificateID CHOICE, would not need an authorityKeyIdentifier extension, as it is explicitly linked to the key in the referred certificate. However, as this profile states (in section 4.2.3), ACs MUST use the v2Form with issuerName CHOICE, this duplication does not arise.

name	id-ce-authorityKeyIdentifier
OID	{ id-ce 35 }
syntax	AuthorityKeyIdentifier
criticality	MUST be FALSE

4.3.4 Authority Information Access

The authorityInformationAccess extension, as defined in [PKIXPROF], MAY be used to assist the AC verifier in checking the revocation status of the AC. Support for the id-ad-caIssuers accessMethod is NOT REQUIRED by this profile since AC chains are not expected.

The following accessMethod is used to indicate that revocation status checking is provided for this AC, using the OCSP protocol defined in [OCSP]:

```
id-ad-ocsp OBJECT IDENTIFIER ::= { id-ad 1 }
```

The accessLocation MUST contain a URI, and the URI MUST contain an HTTP URL [URL] that specifies the location of an OCSP responder. The AC issuer MUST, of course, maintain an OCSP responder at this location.

name	id-ce-authorityInfoAccess
OID	{ id-pe 1 }
syntax	AuthorityInfoAccessSyntax
criticality	MUST be FALSE

4.3.5 CRL Distribution Points

The crlDistributionPoints extension, as profiled in [PKIXPROF], MAY be used to assist the AC verifier in checking the revocation status of the AC. See section 6 for details on revocation.

If the crlDistributionPoints extension is present, then exactly one distribution point MUST be present. The crlDistributionPoints extension MUST use the DistributionPointName option, which MUST contain a fullName, which MUST contain a single name form. That name MUST contain either a distinguished name or a URI. The URI MUST be either an HTTP URL or an LDAP URL [URL].

name	id-ce-cRLDistributionPoints
OID	{ id-ce 31 }
syntax	CRLDistPointsSyntax
criticality	MUST be FALSE

4.3.6 No Revocation Available

The noRevAvail extension, defined in [X.509-2000], allows an AC issuer to indicate that no revocation information will be made available for this AC.

This extension MUST be non-critical. An AC verifier that does not understand this extension might be able to find a revocation list from the AC issuer, but the revocation list will never include an entry for the AC.

name	id-ce-noRevAvail
OID	{ id-ce 56 }
syntax	NULL (i.e. '0500'H is the DER encoding)
criticality	MUST be FALSE

4.4 Attribute Types

Some of the attribute types defined below make use of the IetfAttrSyntax type, also defined below. The reasons for using this type are:

1. It allows a separation between the AC issuer and the attribute policy authority. This is useful for situations where a single policy authority (e.g. an organization) allocates attribute values, but where multiple AC issuers are deployed for performance or other reasons.

2. The syntaxes allowed for values are restricted to OCTET STRING, OBJECT IDENTIFIER, and UTF8String, which significantly reduces the complexity associated with matching more general syntaxes. All multi-valued attributes using this syntax are restricted so that each value MUST use the same choice of value syntax. For example, AC issuers must not use one value with an oid and a second value with a string.

```
IetfAttrSyntax ::= SEQUENCE {
     policyAuthority [0] GeneralNames    OPTIONAL,
     values          SEQUENCE OF CHOICE {
                       octets    OCTET STRING,
                       oid       OBJECT IDENTIFIER,
                       string    UTF8String
                     }
}
```

In the descriptions below, each attribute type is either tagged "Multiple Allowed" or "One Attribute value only; multiple values within the IetfAttrSyntax". This refers to the SET OF AttributeValues; the AttributeType still only occurs once, as specified in section 4.2.7.

4.4.1 Service Authentication Information

The SvceAuthInfo attribute identifies the AC holder to the server/service by a name, and the attribute MAY include optional service specific authentication information. Typically this will contain a username/password pair for a "legacy" application.

This attribute provides information that can be presented by the AC verifier to be interpreted and authenticated by a separate application within the target system. Note that this is a different use to that intended for the accessIdentity attribute in 4.4.2 below.

This attribute type will typically be encrypted when the authInfo field contains sensitive information, such as a password.

```
name      id-aca-authenticationInfo
OID       { id-aca 1 }
Syntax    SvceAuthInfo
values:   Multiple allowed

     SvceAuthInfo ::=    SEQUENCE {
          service   GeneralName,
          ident     GeneralName,
          authInfo  OCTET STRING OPTIONAL
     }
```

4.4.2 Access Identity

The accessIdentity attribute identifies the AC holder to the server/service. For this attribute the authInfo field MUST NOT be present.

This attribute is intended to be used to provide information about the AC holder, that can be used by the AC verifier (or a larger system of which the AC verifier is a component) to authorize the actions of the AC holder within the AC verifier's system. Note that this is a different use to that intended for the svceAuthInfo attribute described in 4.4.1 above.

```
name      id-aca-accessIdentity
OID       { id-aca 2 }
syntax    SvceAuthInfo
values:   Multiple allowed
```

4.4.3 Charging Identity

The chargingIdentity attribute identifies the AC holder for charging purposes. In general, the charging identity will be different from other identities of the holder. For example, the holder's company may be charged for service.

```
name      id-aca-chargingIdentity
OID       { id-aca 3 }
syntax    IetfAttrSyntax
values:   One Attribute value only; multiple values within the
          IetfAttrSyntax
```

4.4.4 Group

The group attribute carries information about group memberships of the AC holder.

```
name      id-aca-group
OID       { id-aca 4 }
syntax    IetfAttrSyntax
values:   One Attribute value only; multiple values within the
          IetfAttrSyntax
```

4.4.5 Role

The role attribute, specified in [X.509-2000], carries information about role allocations of the AC holder.

The syntax used for this attribute is:

```
RoleSyntax ::= SEQUENCE {
        roleAuthority   [0] GeneralNames OPTIONAL,
        roleName        [1] GeneralName
}
```

The roleAuthority field MAY be used to specify the issuing authority for the role specification certificate. There is no requirement that a role specification certificate necessarily exists for the roleAuthority. This differs from [X.500-2000], where the roleAuthority field is assumed to name the issuer of a role specification certificate. For example, to distinguish the administrator role as defined by "Baltimore" from that defined by "SPYRUS", one could put the value "urn:administrator" in the roleName field and the value "Baltimore" or "SPYRUS" in the roleAuthority field.

The roleName field MUST be present, and roleName MUST use the uniformResourceIdentifier CHOICE of the GeneralName.

```
name      id-at-role
OID       { id-at 72 }
syntax    RoleSyntax
values:   Multiple allowed
```

4.4.6 Clearance

The clearance attribute, specified in [X.501-1993], carries clearance (associated with security labeling) information about the AC holder.

The policyId field is used to identify the security policy to which the clearance relates. The policyId indicates the semantics of the classList and securityCategories fields.

This specification includes the classList field exactly as it is specified in [X.501-1993]. Additional security classification values, and their position in the classification hierarchy, may be defined by a security policy as a local matter or by bilateral agreement. The basic security classification hierarchy is, in ascending order: unmarked, unclassified, restricted, confidential, secret, and top-secret.

An organization can develop its own security policy that defines security classification values and their meanings. However, the BIT STRING positions 0 through 5 are reserved for the basic security classification hierarchy.

If present, the SecurityCategory field provides further authorization information. The security policy identified by the policyId field indicates the syntaxes that are allowed to be present in the securityCategories SET. An OBJECT IDENTIFIER identifies each of the allowed syntaxes. When one of these syntaxes is present in the securityCategories SET, the OBJECT IDENTIFIER associated with that syntax is carried in the SecurityCategory.type field.

```
Clearance  ::=  SEQUENCE {
     policyId  [0] OBJECT IDENTIFIER,
     classList [1] ClassList DEFAULT {unclassified},
     securityCategories
               [2] SET OF SecurityCategory OPTIONAL
}

ClassList  ::=  BIT STRING {
     unmarked       (0),
     unclassified   (1),
     restricted     (2)
     confidential   (3),
     secret         (4),
     topSecret      (5)
}

SecurityCategory ::= SEQUENCE {
     type      [0]  IMPLICIT OBJECT IDENTIFIER,
     value     [1]  ANY DEFINED BY type
}

-- This is the same as the original syntax which was defined
-- using the MACRO construct, as follows:
-- SecurityCategory ::= SEQUENCE {
--      type      [0]  IMPLICIT SECURITY-CATEGORY,
--      value     [1]  ANY DEFINED BY type
-- }
--
-- SECURITY-CATEGORY MACRO  ::=
-- BEGIN
-- TYPE NOTATION ::= type | empty
-- VALUE NOTATION ::= value (VALUE OBJECT IDENTIFIER)
-- END
```

```
name      { id-at-clearance }
OID       { joint-iso-ccitt(2) ds(5) module(1)
            selected-attribute-types(5) clearance (55) }
syntax    Clearance - imported from [X.501-1993]
values    Multiple allowed
```

4.5 Profile of AC issuer's PKC

The AC issuer's PKC MUST conform to [PKIXPROF], and the keyUsage extension in the PKC MUST NOT explicitly indicate that the AC issuer's public key cannot be used to validate a digital signature. In order to avoid confusion regarding serial numbers and revocations,

an AC issuer MUST NOT also be a PKC Issuer. That is, an AC issuer cannot be a CA as well. So, the AC issuer's PKC MUST NOT have a basicConstraints extension with the cA BOOLEAN set to TRUE.

5. Attribute Certificate Validation

This section describes a basic set of rules that all valid ACs MUST satisfy. Some additional checks are also described which AC verifiers MAY choose to implement.

To be valid an AC MUST satisfy all of the following:

1. Where the holder uses a PKC to authenticate to the AC verifier, the AC holder's PKC MUST be found, and the entire certification path of that PKC MUST be verified in accordance with [PKIXPROF]. As noted in the security considerations section, if some other authentication scheme is used, AC verifiers need to be very careful mapping the identities (authenticated identity, holder field) involved.

2. The AC signature must be cryptographically correct, and the AC issuer's entire PKC certification path MUST be verified in accordance with [PKIXPROF].

3. The AC issuer's PKC MUST also conform to the profile specified in section 4.5 above.

4. The AC issuer MUST be directly trusted as an AC issuer (by configuration or otherwise).

5. The time for which the AC is being evaluated MUST be within the AC validity. If the evaluation time is equal to either notBeforeTime or notAfterTime, then the AC is timely and this check succeeds. Note that in some applications, the evaluation time MAY not be the same as the current time.

6. The AC targeting check MUST pass as specified in section 4.3.2.

7. If the AC contains an unsupported critical extension, the AC MUST be rejected.

Support for an extension in this context means:

1. The AC verifier MUST be able to parse the extension value.

2. Where the extension value SHOULD cause the AC to be rejected, the AC verifier MUST reject the AC.

Additional Checks:

1. The AC MAY be rejected on the basis of further AC verifier configuration. For example, an AC verifier may be configured to reject ACs which contain or lack certain attributes.

2. If the AC verifier provides an interface that allows applications to query the contents of the AC, then the AC verifier MAY filter the attributes from the AC on the basis of configured information. For example, an AC verifier might be configured not to return certain attributes to certain servers.

6. Revocation

In many environments, the validity period of an AC is less than the time required to issue and distribute revocation information. Therefore, short-lived ACs typically do not require revocation support. However, long-lived ACs and environments where ACs enable high value transactions MAY require revocation support.

Two revocation schemes are defined, and the AC issuer should elect the one that is best suited to the environment in which the AC will be employed.

"Never revoke" scheme:

ACs may be marked so that the relying party understands that no revocation status information will be made available. The noRevAvail extension is defined in section 4.3.6, and the noRevAvail extension MUST be present in the AC to indicate use of this scheme.

Where no noRevAvail is present, the AC issuer is implicitly stating that revocation status checks are supported, and some revocation method MUST be provided to allow AC verifiers to establish the revocation status of the AC.

"Pointer in AC" scheme:

ACs may "point" to sources of revocation status information, using either an authorityInfoAccess extension or a crlDistributionPoints extension within the AC.

For AC users, the "never revoke" scheme MUST be supported, and the "pointer in AC" scheme SHOULD be supported. If only the "never revoke" scheme is supported, then all ACs that do not contain a noRevAvail extension, MUST be rejected.

For AC issuers, the "never revoke" scheme MUST be supported. If all ACs that will ever be issued by that AC issuer, contains a noRevAvail extension, the "pointer in AC" scheme need not be supported. If any AC can be issued that does not contain the noRevAvail extension, the "pointer in AC" scheme MUST be supported.

An AC MUST NOT contain both a noRevAvail and a "pointer in AC".

An AC verifier MAY use any source for AC revocation status information.

7. Optional Features

This section specifies features that MAY be implemented. Conformance to this profile does NOT require support for these features; however, if these features are offered, they MUST be offered as described below.

7.1 Attribute Encryption

Where an AC will be carried in clear within an application protocol or where an AC contains some sensitive information like a legacy application username/password, then encryption of AC attributes MAY be needed.

When a set of attributes are to be encrypted within an AC, the Cryptographic Message Syntax, EnvelopedData structure [CMS] is used to carry the ciphertext and associated per-recipient keying information.

This type of attribute encryption is targeted. Before the AC is signed, the attributes are encrypted for a set of predetermined recipients.

The AC then contains the ciphertext inside its signed data. The EnvelopedData (id-envelopedData) ContentType is used, and the content field will contain the EnvelopedData type.

The ciphertext is included in the AC as the value of an encAttrs attribute. Only one encAttrs attribute can be present in an AC; however, the encAttrs attribute MAY be multi-valued, and each of its values will contain an independent EnvelopedData.

Each value can contain a set of attributes (each possibly a multi-valued attribute) encrypted for a set of predetermined recipients.

The cleartext that is encrypted has the type:

```
ACClearAttrs ::= SEQUENCE {
     acIssuer  GeneralName,
     acSerial  INTEGER,
     attrs     SEQUENCE OF Attribute
}
```

The DER encoding of the ACClearAttrs structure is used as the encryptedContent field of the EnvelopedData. The DER encoding MUST be embedded in an OCTET STRING.

The acIssuer and acSerial fields are present to prevent ciphertext stealing. When an AC verifier has successfully decrypted an encrypted attribute, it MUST then check that the AC issuer and serialNumber fields contain the same values. This prevents a malicious AC issuer from copying ciphertext from another AC (without knowing its corresponding plaintext).

The procedure for an AC issuer when encrypting attributes is illustrated by the following (any other procedure that gives the same result MAY be used):

1. Identify the sets of attributes that are to be encrypted for each set of recipients.
2. For each attribute set which is to be encrypted:
   2.1. Create an EnvelopedData structure for the data for this set of recipients.
   2.2. Encode the ContentInfo containing the EnvelopedData as a value of the encAttrs attribute.
   2.3. Ensure the cleartext attributes are not present in the to-be-signed AC.
3. Add the encAttrs (with its multiple values) to the AC.

Note that there may be more than one attribute of the same type (the same OBJECT IDENTIFIER) after decryption. That is, an AC MAY contain the same attribute type both in clear and in encrypted form (and indeed several times if the same recipient is associated with more than one EnvelopedData). One approach implementers may choose, would be to merge attribute values following decryption in order to re-establish the "once only" constraint.

name	id-aca-encAttrs
OID	{ id-aca 6}
Syntax	ContentInfo
values	Multiple Allowed

If an AC contains attributes, apparently encrypted for the AC verifier, the decryption process MUST not fail. If decryption does fail, the AC MUST be rejected.

7.2 Proxying

When a server acts as a client for another server on behalf of the AC holder, the server MAY need to proxy an AC. Such proxying MAY have to be done under the AC issuer's control, so that not every AC is proxiable and so that a given proxiable AC can be proxied in a targeted fashion. Support for chains of proxies (with more than one intermediate server) MAY also be required. Note that this does not involve a chain of ACs.

In order to meet this requirement we define another extension, ProxyInfo, similar to the targeting extension.

When this extension is present, the AC verifier must check that the entity from which the AC was received was allowed to send it and that the AC is allowed to be used by this verifier.

The proxying information consists of a set of proxy information, each of which is a set of targeting information. If the verifier and the sender of the AC are both named in the same proxy set, the AC can then be accepted (the exact rule is given below).

The effect is that the AC holder can send the AC to any valid target which can then only proxy to targets which are in one of the same proxy sets as itself.

The following data structure is used to represent the targeting/proxying information.

```
ProxyInfo ::= SEQUENCE OF Targets
```

As in the case of targeting, the targetCert CHOICE MUST NOT be used.

A proxy check succeeds if either one of the conditions below is met:

1. The identity of the sender, as established by the underlying authentication service, matches the holder field of the AC, and the current server "matches" any one of the proxy sets. Recall that "matches" is as defined section 4.3.2.

2. The identity of the sender, as established by the underlying authentication service, "matches" one of the proxy sets (call it set "A"), and the current server is one of the targetName fields in the set "A", or the current server is a member of one of the targetGroup fields in set "A".

When an AC is proxied more than once, a number of targets will be on the path from the original client, which is normally, but not always, the AC holder. In such cases, prevention of AC "stealing" requires that the AC verifier MUST check that all targets on the path are members of the same proxy set. It is the responsibility of the AC-using protocol to ensure that a trustworthy list of targets on the path is available to the AC verifier.

name	id-pe-ac-proxying
OID	{ id-pe 10 }
syntax	ProxyInfo
criticality	MUST be TRUE

7.3 Use of ObjectDigestInfo

In some environments, it may be required that the AC is not linked either to an identity (via entityName) or to a PKC (via baseCertificateID). The objectDigestInfo CHOICE in the holder field allows support for this requirement.

If the holder is identified with the objectDigestInfo field, then the AC version field MUST contain v2 (the integer 1).

The idea is to link the AC to an object by placing a hash of that object into the holder field of the AC. For example, this allows production of ACs that are linked to public keys rather than names. It also allows production of ACs which contain privileges associated with an executable object such as a Java class. However, this profile only specifies how to use a hash over a public key or PKC. That is, conformant ACs MUST NOT use the otherObjectTypes value for the digestedObjectType.

To link an AC to a public key, the hash must be calculated over the representation of that public key which would be present in a PKC, specifically, the input for the hash algorithm MUST be the DER encoding of a SubjectPublicKeyInfo representation of the key. Note: This includes the AlgorithmIdentifier as well as the BIT STRING. The rules given in [PKIXPROF] for encoding keys MUST be followed. In this case, the digestedObjectType MUST be publicKey and the otherObjectTypeID field MUST NOT be present.

Note that if the public key value used as input to the hash function has been extracted from a PKC, it is possible that the SubjectPublicKeyInfo from that PKC is NOT the value which should be hashed. This can occur if DSA Dss-parms are inherited as described in section 7.3.3 of [PKIXPROF]. The correct input for hashing in this context will include the value of the parameters inherited from the CA's PKC, and thus may differ from the SubjectPublicKeyInfo present in the PKC.

Implementations which support this feature MUST be able to handle the representations of public keys for the algorithms specified in section 7.3 of [PKIXPROF]. In this case, the digestedObjectType MUST be publicKey and the otherObjectTypeID field MUST NOT be present.

In order to link an AC to a PKC via a digest, the digest MUST be calculated over the DER encoding of the entire PKC, including the signature value. In this case the digestedObjectType MUST be publicKeyCert and the otherObjectTypeID field MUST NOT be present.

7.4 AA Controls

During AC validation a relying party has to answer the question: is this AC issuer trusted to issue ACs containing this attribute? The AAControls PKC extension MAY be used to help answer the question. The AAControls extension is intended to be used in CA and AC issuer PKCs.

```
id-pe-aaControls OBJECT IDENTIFIER ::= { id-pe 6 }

AAControls ::= SEQUENCE {
   pathLenConstraint   INTEGER (0..MAX) OPTIONAL,
   permittedAttrs      [0] AttrSpec OPTIONAL,
   excludedAttrs       [1] AttrSpec OPTIONAL,
   permitUnSpecified   BOOLEAN DEFAULT TRUE
}

AttrSpec::= SEQUENCE OF OBJECT IDENTIFIER
```

The AAControls extension is used as follows:

The pathLenConstraint, if present, is interpreted as in [PKIXPROF]. It restricts the allowed distance between the AA CA (a CA directly trusted to include AAControls in its PKCs), and the AC issuer.

The permittedAttrs field specifies a set of attribute types that any AC issuer below this AA CA is allowed to include in ACs. If this field is not present, it means that no attribute types are explicitly allowed.

The excludedAttrs field specifies a set of attribute types that no AC issuer is allowed to include in ACs. If this field is not present, it means that no attribute types are explicitly disallowed.

The permitUnSpecified field specifies how to handle attribute types which are not present in either the permittedAttrs or excludedAttrs fields. TRUE (the default) means that any unspecified attribute type is allowed in ACs; FALSE means that no unspecified attribute type is allowed.

When AAControls are used, the following additional checks on an AA's PKC chain MUST all succeed for the AC to be valid:

1. Some CA on the ACs certificate path MUST be directly trusted to issue PKCs which precede the AC issuer in the certification path; call this CA the "AA CA".

2. All PKCs on the path from the AA CA, down to and including the AC issuer's PKC, MUST contain an AAControls extension; however, the AA CA's PKC need not contain this extension.

3. Only those attributes in the AC which are allowed, according to all of the AAControls extension values in all of the PKCs from the AA CA to the AC issuer, may be used for authorization decisions; all other attributes MUST be ignored. This check MUST be applied to the set of attributes following attribute decryption, and the id-aca-encAttrs type MUST also be checked.

name	id-pe-aaControls
OID	{ id-pe 6 }
syntax	AAControls
criticality	MAY be TRUE

8. Security Considerations

The protection afforded for private keys is a critical factor in maintaining security. Failure of AC issuers to protect their private keys will permit an attacker to masquerade as them, potentially generating false ACs or revocation status. Existence of bogus ACs and revocation status will undermine confidence in the system. If the compromise is detected, all ACs issued by the AC issuer MUST be revoked. Rebuilding after such a compromise will be problematic, so AC issuers are advised to implement a combination of strong technical measures (e.g., tamper-resistant cryptographic modules) and appropriate management procedures (e.g., separation of duties) to avoid such an incident.

Loss of an AC issuer's private signing key may also be problematic. The AC issuer would not be able to produce revocation status or perform AC renewal. AC issuers are advised to maintain secure backup for signing keys. The security of the key backup procedures is a critical factor in avoiding key compromise.

The availability and freshness of revocation status will affect the degree of assurance that should be placed in a long-lived AC. While long-lived ACs expire naturally, events may occur during its natural lifetime which negate the binding between the AC holder and the attributes. If revocation status is untimely or unavailable, the assurance associated with the binding is clearly reduced.

The binding between an AC holder and attributes cannot be stronger than the cryptographic module implementation and algorithms used to generate the signature. Short key lengths or weak hash algorithms will limit the utility of an AC. AC issuers are encouraged to note advances in cryptology so they can employ strong cryptographic techniques.

Inconsistent application of name comparison rules may result in acceptance of invalid targeted or proxied ACs, or rejection of valid ones. The X.500 series of specifications defines rules for comparing distinguished names. These rules require comparison of strings without regard to case, character set, multi-character white space substrings, or leading and trailing white space. This specification and [PKIXPROF] relaxes these requirements, requiring support for binary comparison at a minimum.

AC issuers MUST encode the distinguished name in the AC holder.entityName field identically to the distinguished name in the holder's PKC. If different encodings are used, implementations of this specification may fail to recognize that the AC and PKC belong to the same entity.

If an attribute certificate is tied to the holder's PKC using the baseCertificateID component of the Holder field and the PKI in use includes a rogue CA with the same issuer name specified in the baseCertificateID component, this rogue CA could issue a PKC to a malicious party, using the same issuer name and serial number as the proper holder's PKC. Then the malicious party could use this PKC in conjunction with the AC. This scenario SHOULD be avoided by properly managing and configuring the PKI so that there cannot be two CAs with the same name. Another alternative is to tie ACs to PKCs using the publicKeyCert type in the ObjectDigestInfo field. Failing this, AC verifiers have to establish (using other means) that the potential collisions cannot actually occur, for example, the CPSs of the CAs involved may make it clear that no such name collisions can occur.

Implementers MUST ensure that following validation of an AC, only attributes that the issuer is trusted to issue are used in authorization decisions. Other attributes, which MAY be present MUST be ignored. Given that the AA controls PKC extension is optional to implement, AC verifiers MUST be provided with this information by other means. Configuration information is a likely alternative means. This becomes very important if an AC verifier trusts more than one AC issuer.

There is often a requirement to map between the authentication supplied by a particular security protocol (e.g. TLS, S/MIME) and the AC holder's identity. If the authentication uses PKCs, then this mapping is straightforward. However, it is envisaged that ACs will also be used in environments where the holder may be authenticated using other means. Implementers SHOULD be very careful in mapping the authenticated identity to the AC holder.

9. IANA Considerations

Attributes and attribute certificate extensions are identified by object identifiers (OIDs). Many of the OIDs used in this document are copied from X.509 [X.509-2000]. Other OIDs were assigned from an arc delegated by the IANA. No further action by the IANA is necessary for this document or any anticipated updates.

10. References

[CMC] Myers, M., Liu, X., Schaad, J. and J. Weinstein, "Certificate Management Messages over CMS", RFC 2797, April 2000.

[CMP] Adams, C. and S. Farrell, "Internet X.509 Public Key Infrastructure - Certificate Management Protocols", RFC 2510, March 1999.

[CMS] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[ESS] Hoffman, P., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.

[KRB] Kohl, J. and C. Neuman, "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.

[LDAP] Wahl, M., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[OCSP] Myers, M., Ankney, R., Malpani, A., Galperin, S. and C. Adams, "X.509 Internet Public Key Infrastructure - Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.

[PKIXALGS] Bassham, L., Polk, W. and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation Lists CRL Profile", RFC 3279, April 2002.

[PKIXPROF] Housley, R., Polk, T, Ford, W. and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[URL] Berners-Lee, T., Masinter L. and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[X.208-1988] CCITT Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1). 1988.

[X.209-88] CCITT Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). 1988.

[X.501-88] CCITT Recommendation X.501: The Directory - Models. 1988.

[X.501-1993] ITU-T Recommendation X.501 : Information Technology - Open Systems Interconnection - The Directory: Models, 1993.

[X.509-1988] CCITT Recommendation X.509: The Directory - Authentication Framework. 1988.

[X.509-1997] ITU-T Recommendation X.509: The Directory - Authentication Framework. 1997.

[X.509-2000] ITU-T Recommendation X.509: The Directory - Public-Key and Attribute Certificate Frameworks. 2000

Appendix A: Object Identifiers

This (normative) appendix lists the new object identifiers which are defined in this specification. Some of these are required only for support of optional features and are not required for conformance to this profile. This specification mandates support for OIDs which have arc elements with values that are less than 2^32, (i.e. they MUST be between 0 and 4,294,967,295 inclusive) and SHOULD be less than 2^31 (i.e. less than or equal to 2,147,483,647). This allows each arc element to be represented within a single 32 bit word. Implementations MUST also support OIDs where the length of the dotted decimal (see [LDAP], section 4.1.2) string representation can be up to 100 bytes (inclusive). Implementations MUST be able to handle OIDs with up to 20 elements (inclusive). AA's SHOULD NOT issue ACs which contain OIDs that breach these requirements.

The following OIDs are imported from [PKIXPROF]:

```
id-pkix OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
          dod(6) internet(1) security(5) mechanisms(5) pkix(7) }
id-mod  OBJECT IDENTIFIER ::= { id-pkix 0 }
id-pe   OBJECT IDENTIFIER ::= { id-pkix 1 }
id-ad   OBJECT IDENTIFIER ::= { id-pkix 48 }
id-at   OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 4 }
id-ce   OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 29 }
```

The following new ASN.1 module OID is defined:

```
id-mod-attribute-cert        OBJECT IDENTIFIER ::= { id-mod 12 }
```

The following AC extension OIDs are defined:

```
id-pe-ac-auditIdentity       OBJECT IDENTIFIER ::= { id-pe 4 }
id-pe-ac-proxying            OBJECT IDENTIFIER ::= { id-pe 10 }
id-ce-targetInformation      OBJECT IDENTIFIER ::= { id-ce 55 }
```

The following PKC extension OIDs are defined:

```
id-pe-aaControls             OBJECT IDENTIFIER ::= { id-pe 6 }
```

The following attribute OIDs are defined:

```
id-aca                        OBJECT IDENTIFIER ::= { id-pkix 10 }
id-aca-authenticationInfo     OBJECT IDENTIFIER ::= { id-aca 1 }
id-aca-accessIdentity         OBJECT IDENTIFIER ::= { id-aca 2 }
id-aca-chargingIdentity       OBJECT IDENTIFIER ::= { id-aca 3 }
id-aca-group                  OBJECT IDENTIFIER ::= { id-aca 4 }
id-aca-encAttrs               OBJECT IDENTIFIER ::= { id-aca 6 }
id-at-role                    OBJECT IDENTIFIER ::= { id-at 72 }
id-at-clearance               OBJECT IDENTIFIER ::=
            { joint-iso-ccitt(2) ds(5) module(1)
              selected-attribute-types(5) clearance (55) }
```

Appendix B: ASN.1 Module

```
PKIXAttributeCertificate {iso(1) identified-organization(3) dod(6)
             internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
             id-mod-attribute-cert(12)}

   DEFINITIONS IMPLICIT TAGS ::=

   BEGIN

   -- EXPORTS ALL --

   IMPORTS

         -- IMPORTed module OIDs MAY change if [PKIXPROF] changes
         -- PKIX Certificate Extensions
            Attribute, AlgorithmIdentifier, CertificateSerialNumber,
            Extensions, UniqueIdentifier,
            id-pkix, id-pe, id-kp, id-ad, id-at
            FROM PKIX1Explicit88 {iso(1) identified-organization(3)
                     dod(6) internet(1) security(5) mechanisms(5)
                     pkix(7) id-mod(0) id-pkix1-explicit-88(1)}

            GeneralName, GeneralNames, id-ce
            FROM PKIX1Implicit88 {iso(1) identified-organization(3)
                     dod(6) internet(1) security(5) mechanisms(5)
                     pkix(7) id-mod(0) id-pkix1-implicit-88(2)} ;

   id-pe-ac-auditIdentity     OBJECT IDENTIFIER ::= { id-pe 4 }
   id-pe-aaControls           OBJECT IDENTIFIER ::= { id-pe 6 }
   id-pe-ac-proxying          OBJECT IDENTIFIER ::= { id-pe 10 }
   id-ce-targetInformation    OBJECT IDENTIFIER ::= { id-ce 55 }

   id-aca                     OBJECT IDENTIFIER ::= { id-pkix 10 }
```

```
id-aca-authenticationInfo    OBJECT IDENTIFIER ::= { id-aca 1 }
id-aca-accessIdentity        OBJECT IDENTIFIER ::= { id-aca 2 }
id-aca-chargingIdentity      OBJECT IDENTIFIER ::= { id-aca 3 }
id-aca-group                 OBJECT IDENTIFIER ::= { id-aca 4 }
-- { id-aca 5 } is reserved
id-aca-encAttrs              OBJECT IDENTIFIER ::= { id-aca 6 }

id-at-role                   OBJECT IDENTIFIER ::= { id-at 72}
id-at-clearance              OBJECT IDENTIFIER ::=
            { joint-iso-ccitt(2) ds(5) module(1)
              selected-attribute-types(5) clearance (55) }

       -- Uncomment this if using a 1988 level ASN.1 compiler
       -- UTF8String ::= [UNIVERSAL 12] IMPLICIT OCTET STRING

       AttributeCertificate ::= SEQUENCE {
             acinfo               AttributeCertificateInfo,
             signatureAlgorithm   AlgorithmIdentifier,
             signatureValue       BIT STRING
       }

       AttributeCertificateInfo ::= SEQUENCE {
          version        AttCertVersion  -- version is v2,
          holder         Holder,
          issuer         AttCertIssuer,
          signature      AlgorithmIdentifier,
          serialNumber   CertificateSerialNumber,
          attrCertValidityPeriod   AttCertValidityPeriod,
          attributes     SEQUENCE OF Attribute,
          issuerUniqueID UniqueIdentifier OPTIONAL,
          extensions     Extensions     OPTIONAL
       }

       AttCertVersion ::= INTEGER { v2(1) }

       Holder ::= SEQUENCE {
             baseCertificateID   [0] IssuerSerial OPTIONAL,
                       -- the issuer and serial number of
                       -- the holder's Public Key Certificate
             entityName          [1] GeneralNames OPTIONAL,
                       -- the name of the claimant or role
             objectDigestInfo    [2] ObjectDigestInfo OPTIONAL
                       -- used to directly authenticate the
                       -- holder, for example, an executable
       }
```

```
ObjectDigestInfo    ::= SEQUENCE {
      digestedObjectType  ENUMERATED {
           publicKey            (0),
           publicKeyCert        (1),
           otherObjectTypes     (2) },
                   -- otherObjectTypes MUST NOT
                   -- MUST NOT be used in this profile
      otherObjectTypeID   OBJECT IDENTIFIER  OPTIONAL,
      digestAlgorithm     AlgorithmIdentifier,
      objectDigest        BIT STRING
}

AttCertIssuer ::= CHOICE {
      v1Form   GeneralNames,  -- MUST NOT be used in this
                              -- profile
      v2Form   [0] V2Form     -- v2 only
}

V2Form ::= SEQUENCE {
      issuerName            GeneralNames  OPTIONAL,
      baseCertificateID     [0] IssuerSerial  OPTIONAL,
      objectDigestInfo      [1] ObjectDigestInfo  OPTIONAL
         -- issuerName MUST be present in this profile
         -- baseCertificateID and objectDigestInfo MUST
         -- NOT be present in this profile
}

IssuerSerial  ::=  SEQUENCE {
      issuer         GeneralNames,
      serial         CertificateSerialNumber,
      issuerUID      UniqueIdentifier OPTIONAL
}

AttCertValidityPeriod  ::= SEQUENCE {
      notBeforeTime  GeneralizedTime,
      notAfterTime   GeneralizedTime
}

Targets ::= SEQUENCE OF Target

Target  ::= CHOICE {
      targetName     [0] GeneralName,
      targetGroup    [1] GeneralName,
      targetCert     [2] TargetCert
}
```

```
TargetCert  ::= SEQUENCE {
      targetCertificate  IssuerSerial,
      targetName         GeneralName OPTIONAL,
      certDigestInfo     ObjectDigestInfo OPTIONAL
}

IetfAttrSyntax ::= SEQUENCE {
     policyAuthority[0] GeneralNames    OPTIONAL,
     values         SEQUENCE OF CHOICE {
                    octets    OCTET STRING,
                    oid       OBJECT IDENTIFIER,
                    string    UTF8String
    }
}

SvceAuthInfo ::=    SEQUENCE {
      service       GeneralName,
      ident         GeneralName,
      authInfo      OCTET STRING OPTIONAL
}

RoleSyntax ::= SEQUENCE {
      roleAuthority  [0] GeneralNames OPTIONAL,
      roleName       [1] GeneralName
}

Clearance  ::=  SEQUENCE {
      policyId       [0] OBJECT IDENTIFIER,
      classList      [1] ClassList DEFAULT {unclassified},
      securityCategories
                     [2] SET OF SecurityCategory  OPTIONAL
}

ClassList  ::=  BIT STRING {
      unmarked       (0),
      unclassified   (1),
      restricted     (2),
      confidential   (3),
      secret         (4),
      topSecret      (5)
}

SecurityCategory ::= SEQUENCE {
      type      [0]  IMPLICIT OBJECT IDENTIFIER,
      value     [1]  ANY DEFINED BY type
}
```

```
            AAControls ::= SEQUENCE {
                  pathLenConstraint INTEGER (0..MAX) OPTIONAL,
                  permittedAttrs    [0] AttrSpec OPTIONAL,
                  excludedAttrs     [1] AttrSpec OPTIONAL,
                  permitUnSpecified BOOLEAN DEFAULT TRUE
            }

            AttrSpec::= SEQUENCE OF OBJECT IDENTIFIER

            ACClearAttrs ::= SEQUENCE {
                  acIssuer          GeneralName,
                  acSerial          INTEGER,
                  attrs             SEQUENCE OF Attribute
            }

            ProxyInfo ::= SEQUENCE OF Targets

      END
```

Author's Addresses

Stephen Farrell
Baltimore Technologies
39/41 Parkgate Street
Dublin 8
IRELAND

EMail: stephen.farrell@baltimore.ie

Russell Housley
RSA Laboratories
918 Spring Knoll Drive
Herndon, VA 20170
USA

EMail: rhousley@rsasecurity.com

Acknowledgements

Russ Housley thanks the management at SPYRUS, who supported the development of this specification while he was employed at SPYRUS. Russ Housley also thanks the management at RSA Laboratories, who supported the completion of the specification after a job change.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.