PKIX Working Group
INTERNET-DRAFT
Expires in six months

S. Farrell
Baltimore Technologies
R. Housley
SPYRUS
March 2000

An Internet Attribute Certificate Profile for Authorization

<draft-ietf-pkix-ac509prof-02.txt>

Status of this Memo

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

Status of this Memo.....1

Abstract....1
Table of Contents....1
1. Introduction....3
1.1 Delegation and AC chains....4

1.2 Attribute Certificate Distribution ("push" vs "pull")..4
1.3 Document Structure....6
2. Terminology....7
3. Requirements....8
4. The AC Profile....9
4.1 X.509 Attribute Certificate Definition....9
4.2 Profile of Standard Fields....11
4.2.1 Version....11
4.2.2 Holder....11
4.2.3 Issuer....12
4.2.4 Signature....12
4.2.5 Serial Number....13
4.2.6 Validity Period....13
4.2.7 Attributes....13
4.2.8 Issuer Unique Identifier....14
4.2.9 Extensions....14
4.3 Extensions....14
4.3.1 Audit Identity....14
4.3.2 AC Targeting....15
4.3.3 Authority Key Identifier....16
4.3.4 Authority Information Access....17
4.3.5 CRL Distribution Points....17
4.3.6 No Revocation Available....17
4.4 Attribute Types....18
4.4.1 Service Authentication Information....18
4.4.2 Access Identity....19
4.4.3 Charging Identity....19
4.4.4 Group....19
4.4.5 Role....19
4.4.6 Clearance....20
4.5 Profile of AC Issuer's PKC....21
5. Attribute Certificate Validation....22
6. Revocation....23
7. Optional Features....24
7.1 Attribute Encryption....24
7.2 Proxying....25
7.3 Use of ObjectDigestInfo....26
7.4 AA Controls....27
8. Security Considerations....29

9. References.....30
Author's Addresses.....31
Full Copyright Statement.....31
Appendix B: Object Identifiers.....32
Appendix B: "Compilable" ASN.1 Module.....33

1. Introduction

The key words "MUST", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in [RFC2119].

A server makes an access control decision when a client requests access to a resource offered by that server. The server must ensure that the client is authorized to access that resource. The server decision is based on the access control policy, the context of the request, and the identity and authorizations of the client. The access control policy and the context of the request are readily available to the server. Certificates may be used to provide identity and authorization information about the client.

Similar access control decisions are made in other network environments, such as a store-and-forward electronic mail environment. That is, access control decisions are not limited to client-server protocol environments.

X.509 public key certificates (PKCs) [X.509-97], [X.509-DAM], [PKIXPROF] bind an identity and a public key. The identity may be used to support identity-based access control decisions after the client proves that it has access to the private key that corresponds to the public key contained in the PKC. The public key is used to validate digital signatures or cryptographic key management operations. However, not all access control decisions are identity-based. Rule-based, role-based, and rank-based access control decisions require additional information. For example, information about a client's ability to pay for a resource access may be more important than the client's identity. Authorization information to support such access control decisions 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 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 this authorization information from the PKC. Yet, this authorization information also needs to be protected in a fashion similar to a PKC. An attribute certificate (AC) provides this protection, and it is simply a digitally signed (or certified) set of attributes.

An AC is a structure similar to a PKC; the main difference being that it contains no public key. An AC may contain attributes that specify group membership, role, security clearance, and other access control information associated with the AC holder. The syntax for

the AC is defined in Recommendation X.509 (making the term "X.509 certificate" ambiguous). This document specifies a profile of the X.509 AC suitable for use with authorization information within Internet protocols.

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. For example, one way in which the linkage between the request and the AC can be achieved is if the AC has a "pointer" to a PKC for the requester and that PKC has been used to authenticate the access request.

As there is often confusion about the difference between PKCs and ACs, an analogy may help. 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.

1.1 Delegation and AC chains

The X.509 standard defines delegation as "Conveyance of privilege from one entity that holds such privilege, to another entity". It further defines a delegation path as "An ordered sequence of certificates which, together with authentication of a privilege

asserter's identity can be processed to verify the authenticity of a privilege asserter's privilege". It then goes on to define various mechanisms for use in delegation cases involving "chains" of ACs.

As the administration and processing associated with such AC chains is potentially much more complex than use of a single AC, and as the use of ACs in the Internet is today in its infancy, this specification does not address such AC chains. Other (future) specifications may address the use of AC chains.

This means that conformant implementations are only REQUIRED to be able to "handle" a single AC at a time. Note however, that validation of that AC MAY require validation of a full PKC chain, 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 access control decision functions. However, there are a number of possible communication paths that an AC may take.

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.

In other cases, it is more suitable for a client simply to authenticate to the server and for the server to request ("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 client-server protocol. It is also more suitable for some inter-domain cases where the client's rights should be assigned within the server's domain, rather than within the client's "home" domain.

There are a number of possible exchanges that can occur and three entities involved (client, server and AC issuer). In addition the use of 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 protocol for these exchanges.

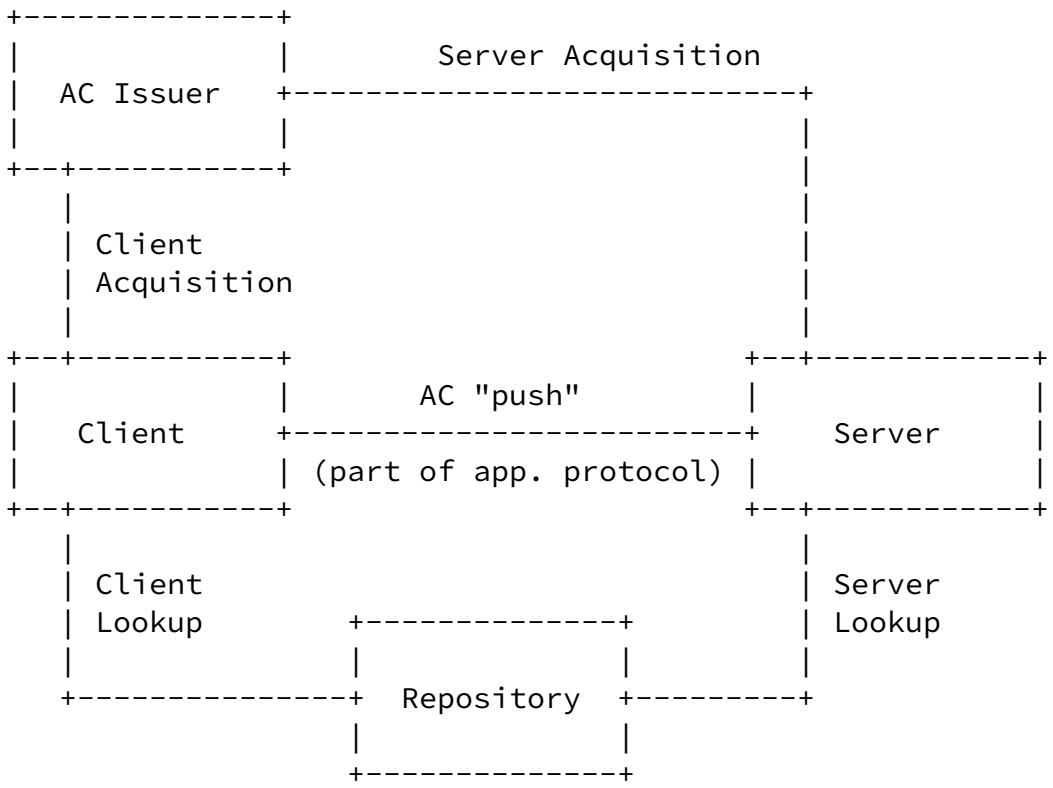

Figure 1: AC Exchanges

1.3 Document Structure

The remainder of the document is structured as follows:-

Section 2 defines some terminology
Section 3 specifies the requirements that this profile is 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 but for which support is not required for conformance to this profile

Appendices contain the list of OIDs required to support this specification and a "compilable" 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, e.g. in the S/MIME v3 context, the mail user agent would, by turns, be both "client" and "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 Attribute Certificate profile meets the following requirements.

Time/Validity requirements:

1. Support for short-lived or long-lived ACs is required. Typical validity periods might be measured in hours, as opposed to months for X.509 public key certificates. Short validity periods mean that ACs can 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 should be defined which can be contained within ACs, for example "access identity", "group", "role", "clearance", "audit identity", "charging id" etc.
4. Standard attribute types should be defined so that it is possible for an AC verifier to distinguish between e.g. the "Administrators group" as defined by Baltimore and the "Administrators group" as defined by SPYRUS.

Targeting of ACs:

5. It should be possible to "target" an AC. This means that a given AC may be "targeted" at one, or a small number of, servers in the sense that a trustworthy non- target 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 (which may be an online AC issuer).

4. The AC Profile

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 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 attribute certificates 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; as has been done for PKCs [PKIXPROF], this document uses the 1988 ASN.1 syntax, the encoded certificate and standard extensions are equivalent.

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 Attribute Certificate given below. Types that are not defined can be found in [PKIXPROF].

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

AttributeCertificateInfo ::= SEQUENCE {
     version        AttCertVersion DEFAULT v1,
     holder         Holder,
     issuer         AttCertIssuer,
     signature      AlgorithmIdentifier,
     serialNumber   CertificateSerialNumber,
     attrCertValidityPeriod   AttCertValidityPeriod,
     attributes     SEQUENCE OF Attribute,
     issuerUniqueID UniqueIdentifier OPTIONAL,
     extensions     Extensions     OPTIONAL
}

AttCertVersion ::= INTEGER {v1(0), 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
               -- if present, version must be v2
}

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 {
        oldForm         GeneralNames,
        newForm    [0] SEQUENCE {
        issuerName               GeneralNames      OPTIONAL,
        baseCertificateId   [0]  IssuerSerial      OPTIONAL,
        objectDigestInfo    [1]  ObjectDigestInfo  OPTIONAL
         --at least one of issuerName, baseCertificateId or --
         -- objectDigestInfo must be present --
         -- if newForm is used, version must be v2--
    }

    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
```

Implementers should note that the DER encoding (DER is defined in [X.208-88]) 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, its 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

For all GeneralName fields in this profile the otherName, x400Address, ediPartyName and registeredID options MUST NOT be used unless otherwise specified (e.g. as in the description of targeting extension).

The use of Kerberos [KRB] format names, encoded into the otherName, SHOULD however, be supported using the krb5PrincipalName OID and the KerberosName syntax as defined in [PKINIT].

This means that unless otherwise indicated,(e.g. for the role attribute), conforming implementations MUST be able to support the dNSName, directoryName, uniformResourceIdentifier and iPAddress fields in all cases where GeneralName is used. The MUST support requirements for each of these fields are as specified in [PKIXPROF], (mainly in section 4.2.1.7).

4.2.1 Version

This MUST be the default value of v1, i.e. not present in the DER encoding, except where the holder is identified using the optional objectDigestInfo field, as specified in section 7.3.

4.2.2 Holder

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 public key certificate (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 holder.baseCertificateID.issuerUID field MUST only be used if the holder's PKC contains an issuerUniqueID field (in which case, the same value MUST be present in the holder.baseCertificateID.issuerUID_field). 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, then the entityName MUST be the same as the PKC subject field, unless the PKC subject field contains an empty distinguished name. In that case, the entityName field MUST be identical to one of the values of the PKC subjectAltName field extension. Note that [PKIXPROF] mandates that the subjectAltNames extension be present if the PKC subject is a non empty distinguished name.

In any other case where the holder field uses the entityName option then 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 define in that protocol.

4.2.3 Issuer

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

Part of the reason for the use of the issuerName field is that it allows the AC verifier to be independent of the AC issuer's public key infrastructure. Using the baseCertificateId field to reference the AC issuer would mean that the AC verifier would have such a dependency.

4.2.4 Signature

Contains the algorithm identifier used to validate the AC signature.

This MUST be one of the following algorithms defined in [PKIXPROF] section 7.2: md5WithRSAEncryption, id-dsa-with-sha1 or sha-

1WithRSAEncryption, or ecdsa-with-SHA1 defined in [ECDSA] section 3.2.

id-dsa-with-sha1 MUST be supported by all AC users. The other algorithms SHOULD be supported.

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 (one second is the shortest possible validity due to the use of GeneralizedTime).

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 32 bits. Conformant ACs MUST NOT use 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, but they MUST be unique for a given AC issuer.

4.2.6 Validity Period

The attrCertValidityPeriod (a.k.a. validity) field specifies the period for which the AC issuer expects 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. Optionally, the GeneralizedTime field can include a representation of the time differential between local and Greenwich Mean Time.

For the purposes of this profile, GeneralizedTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero. GeneralizedTime values MUST NOT include fractional seconds.

(Note that the above is as specified in [PKIXPROF], section 4.1.2.5.2.)

Note that AC users MUST be able to handle the case where an AC is issued, which (at the time of parsing), has its entire validity period in the future (a "post-dated" AC). This is valid for some applications, e.g. 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 attribute type OID in the sequence MUST be unique, that is, only one instance of each attribute can occur in a single AC. All instances can however, be multi-valued.

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

Note that 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 4.5.

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.

Section 4.3 defines the extensions that MAY be used with this

profile. An AC that has no extensions conforms to the profile. If any other critical extension is used, then the AC does not conform to this profile. An AC that contains additional non-critical extensions still conforms.

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 certificates 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 do not contain records which directly identify individuals. This may make the use of the holder field of the AC unsuitable for use in audit trails.

In order 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 except with the co-operation of the AC issuer.

The value of the audit identity plus the AC issuer/serial SHOULD then be used for audit/logging purposes. If the value of the audit identity is suitably chosen then 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 map "backwards" from the audit identity to the actual identity of the AC holder.

Of course, auditing could be based on the AC issuer/serial pair, however, this method doesn't allow tracking the same AC holder across different ACs. This means that an audit identity is only useful if it lasts for longer than the typical AC lifetime - how much longer is an issue for the AC issuer implementation. Auditing could also be based on the AC holder's PKC issuer/serial however, this will often allow the server/service administrator identify the AC holder.

As the AC verifier might otherwise use the AC subject 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 are "semi-anonymous".

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

In order to allow that an AC is "targeted", the target information extension 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 then 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,
     targetCertificate   [2] IssuerSerial,
     targetDigest        [3] ObjectDigestInfo
}
```

The targetCertificate and targetDigest fields are only present to allow future compatibility with [X.509-DAM] and MUST NOT be used.

The targets check passes if the current server (recipient) is one of the targetName fields in the targets part, or, the current server is a member of one of the targetGroup fields in the targets part. 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 targetGroup's to which it belongs or can otherwise determine its membership. For example, if the targetGroup were to be a DNS domain and the AC verifier knows the DNS domain to which it belongs. Another example would be where the targetGroup is "PRINTERS" and the AC verifier "knows" that it's a printer or print server.

```
   name            id-pe-ac-targeting
<<will change this to id-ce-targetInformation from DAM if ISO can
change its syntax otherwise stick with the PKIX OID and live with
two different targeting extensions>>
   OID             { id-pe 5 }
   syntax          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) that ACs MUST use the entityName choice, this does not arise here.

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. Note that support for the id-ad-caIssuers accessMethod defined in [PKIXPROF] is NOT REQUIRED as in this profile, the authorityInformationAccess is only used for revocation status checking. Conformant ACs containing this extension MUST contain exactly one AccessDescription.

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, this MUST contain an HTTP URL, specifying 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 on revocation below for details.

Exactly one distribution point MUST be present, it MUST use the DistributionPointName option, which MUST contain a fullName, which MUST contain a single name form. That name MUST contain either an HTTP URL or a distinguished name.

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

4.3.6 No Revocation Available

This extension (imported from [X.509-DAM]) allows an AC issuer to indicate that no revocation information will be made available for this AC.

This extension MUST be non-critical, on the basis that an AC verifier that does not understand it can still find a revocation list (for example), but won't ever find 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 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, network or other reasons.
2. The syntaxes allowed for values are restricted to OCTET STRING OID and UTF8String, which reduces some of the matching complexities associated with more general syntaxes. All multi-valued attributes using this syntax are restricted so that each value MUST use the same choice of value syntax, that is, it is not allowed that one value use an OID but that a second value uses 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 tagged as either "Multiple Allowed" or "One Attribute value only; multiple values within the IetfAttrSyntax". This refers to the SET OF AttributeValue, the AttributeType still only occurs once, as specified in section 4.2.7.

4.4.1 Service Authentication Information

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

This attribute type will typically need to be encrypted if the authInfo field contains sensitive information (e.g. 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

An access identity identifies the AC holder to the server/service. For this attribute the authInfo field MUST NOT be present.

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

4.4.3 Charging Identity

This attribute type identifies the AC holder for charging purposes. Note that, in general, the charging identity will be different from other identities of the holder, for example, when the holderÆs company is to 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

This 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

This attribute (imported from [X.509-DAM]) 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 MUST NOT be used. The roleName field MUST be present and MUST use the uniformResourceIdentifier field of the GeneralName.

```
name      id-at-role
OID       { id-aca 5 }
syntax    RoleSyntax
values:   Multiple allowed
```

4.4.6 Clearance

This attribute (imported from [X.501]) carries clearance (security labeling) information about the AC holder.

```
Clearance  ::=  SEQUENCE {
     policyId  OBJECT IDENTIFIER,
     classList ClassList DEFAULT {unclassified},
     securityCategories
               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
```


The security category value above can uses the ASN.1 ANY construct. Conformant ACs MUST only use UTF8String, OID and OCTET STRING syntaxes for this value.

```
   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]
values    Multiple allowed
```

4.5 Profile of AC Issuer's PKC

The AC Issuer's PKC MUST conform to [PKIXPROF] and its keyUsage MUST NOT explicitly indicate that the AC issuer can't sign. In order to avoid confusion (e.g. over serial numbers or revocations) an AC issuer MUST NOT also be a PKC Issuer (i.e. it can't be a CA as well), so the AC Issuer's PKC MUST NOT have a basicConstraints extension with isACA 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. The AC signature must be cryptographically correct and the AC issuer's entire certification path (including the AC issuer's PKC) MUST be verified in accordance with [PKIXPROF].
2. The AC issuer's PKC MUST also conform to the profile specified in section 4.5 above.
3. The AC issuer MUST be directly trusted as an AC issuer (by configuration or otherwise).
4. 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, i.e. this check succeeds). Note that in some applications, the evaluation time MAY not be the same as the current time.
5. The AC targeting check MUST pass (see section 4.3.2 above)
6. If the AC contains any "unsupported" critical extensions then the AC MUST be rejected.

"Support" for an extension in this context means:

a. the AC verifier MUST be able to parse the extension value, and,
b. 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 attribute types.
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, e.g. an AC verifier might be configured not to return certain attributes to certain targets.

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.

The basic approach taken is to allow use of the following AC revocation related schemes.

"Never revoke" scheme: ACs may be marked so that the relying party understands that no revocation status information will be made available. A noRevAvail extension as defined in section 4.3.6 above MUST be present in the AC to indicate this.

Where no noRevAvail is not present, then the AC issuer is implicitly stating that revocation status checks are supported and some method MUST be provided to allow AC verifiers to establish the revocation status of the AC.

"Pointer in AC" scheme: ACs may be marked (like PKCs) to "point" to sources of revocation status information (using an authorityInfoAccess or crlDistributionPoints extension in the AC itself).

For AC users, the "never revoke" scheme MUST be supported, 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, will contain a noRevAvail extension, then the "pointer in AC" scheme NEED NOT be supported. If any AC can be issued that does not contain the noRevAvail extension, then the "pointer in AC" scheme MUST be supported.

All conformant ACs MUST contain exactly one of the noRevAvail, authorityInformationAccess or crlDistributionPoints extensions. That is, the crlDistributionPoints, authorityInformationAccess and noRevAvail extensions are mutually exclusive for a single AC and an AC MUST NOT contain more than one of these extensions. This differs from the case with PKCs. An AC verifier MAY use other (e.g. configured) sources for AC revocation status information.

7. Optional Features

This section specifies features that MAY be implemented. Conformance to this specification does NOT require support for these features.

7.1 Attribute Encryption

Where an AC will be carried in clear within an application protocol or where an AC contains some sensitive information (e.g. 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(s) and associated per-recipient keying information.

This type of attribute encryption is targeted, which means that before the AC is signed the attributes have been encrypted for a set of predetermined recipients.

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

The set of ciphertexts is included into the AC as the value of an encrypted attributes attribute. Only one encrypted attributes

attribute can be present in an AC - however it MAY be multi-valued and each of its values will contain an EnvelopedData.

Each value can contain a set of attributes (each possibly a multi-valued attribute) encrypted for a set of 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, i.e. 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 issuer's AC into an AC issued by the malicious AC issuer.

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 EnvelopedData as a value of the EncryptedAttributes attribute
   2.3. Ensure the cleartext attribute(s) are not present in the to-be-signed AC
3. Add the EncryptedAttribute (with its multiple values) to the AC

Note that the rule that each attribute type (the OID) only occurs once may not hold after decryption. That is, an AC MAY contain the same attribute type both in clear and in encrypted form (and indeed more than once if the decryptor is a recipient for more than one EnvelopedData). One approach implementers may choose, would be to merge attributes 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 then the decryption process MUST not fail - if decryption fails then the AC MUST be rejected.

7.2 Proxying

In some circumstances, a server needs to proxy an AC when it acts as a client (for another server) on behalf of the AC holder. Such proxying may have to be 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) is also sometimes 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 then the AC can 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 targetCertificate and targetDigest fields 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 (where "matches" is as defined for the targeting check in section 4.3.2 above).

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".

Where 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 7 }
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 via the objectDigestInfo field then the AC version field MUST contain v2 (i.e. the integer 1).

The basic 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, or production of ACs which contain privileges associated with an executable object (e.g. 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.

In order 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] and [ECDSA] 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, then it is possible that the SubjectPublicKeyInfo from that PKC is NOT the value which should be hashed. This can occur if, e.g. 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 keys for the algorithms specified in section 7.3 of [PKIXPROF] and those specified in [ECDSA]. 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 (i.e. including the signature bits). 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, intended to be used in CA and AC Issuer PKCs, MAY be used to help answer the question.

Note that this extension is quite likely to change in future based

on experience with the use of ACs in the Internet.

```
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], but now 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 (though the permitUnSpecified field may open things up).

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 (though the permitUnSpecified field may close things down).

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.

Where aaControls are used then the following additional checks on an AA's PKC chain MUST all succeed for the AC to be valid:

1. Some CA on the AC's 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 PKC's on the path from the AA CA down to and including the

AC issuer's PKC MUST contain an aaControls extension as defined below (the PKC with the AA CA's as subject 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 (note that this check MUST be applied to the set of attributes following attribute decryption and that in such cases 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

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, this means that AC verifiers MUST be provided with the required information by other means - e.g. by configuration. This becomes very important if an AC verified trusts more than one AC issuer.

There is often a requirement to map between the authentication supplied by a particular 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. References

[CMC] Myers, M., et al. "Certificate Management Messages over CMS", draft-ietf-pkix-cmc-05.txt, July 1999.

[CMP] Adams, C., Farrell, S., "Internet X.509 Public Key Infrastructure - Certificate Management Protocols", RFC2510.

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

[ESS] Hoffman, P., "Enhanced Security Services for S/MIME", RFC2634.

[ECDSA] D. Johnson, W. Polk, "Internet X.509 Public Key Infrastructure Representation of Elliptic Curve Digital Signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public Key Infrastructure Certificates" draft-ietf-pkix-ipki-ecdsa-02.txt, October 1999.

[LDAP] Wahl, M., et al., "Lightweight Directory Access Protocol (v3)", RFC 2251.

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

[PKINIT] Tung, B., et al., "Public Key Cryptography for Initial Authentication in Kerberos", draft-ietf-cat-kerberos-pk-init-10.txt

[PKIXPROF] Housley, R., Ford, W., Polk, T, & Solo, D., "Internet Public Key Infrastructure - X.509 Certificate and CRL profile",RFC 2459.

[OCSP] Myers, M., et al., " X.509 Internet Public Key Infrastructure - Online Certificate Status Protocol - OCSP", RFC 2560.

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

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

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

[X.208-88] 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.509-88] CCITT Recommendation X.509: The Directory - Authentication Framework. 1988.

[X.509-97] ITU-T Recommendation X.509: The Directory - Authentication Framework. 1997.

[X.509-DAM] ISO 9594-8 Information Technology - Open systems Interconnection - The Directory: Authentication Framework - Draft Amendment 1: Certificate Extensions, October 1999.

Author's Addresses

Stephen Farrell,
Baltimore Technologies
61/62 Fitzwilliam Lane,
Dublin 2,
IRELAND

tel: +353-1-647-3000
email: stephen.farrell@baltimore.ie

Russell Housley,
SPYRUS,
381 Elden Street,
Suite 1120,
Herndon, VA 20170,
USA

email: housley@spyrus.com

Full Copyright Statement

Appendix B: 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 2^28, i.e. they MUST be between 0 and 268,435,455 inclusive. 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 }
```

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-targeting           OBJECT IDENTIFIER ::= { id-pe 5 }
id-pe-ac-proxying            OBJECT IDENTIFIER ::= { id-pe 7 }
```

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 }
```

Appendix B: "Compilable" 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 EXPLICIT TAGS ::=

      BEGIN

      -- EXPORTS ALL --

      IMPORTS

            -- PKIX Certificate Extensions
               Attribute, AlgorithmIdentifier, CertificateSerialNumber,
               Extensions, UniqueIdentifier,
               id-pkix, id-pe, id-kp, id-ad
               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
               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-ac-targeting           OBJECT IDENTIFIER ::= { id-pe 5 }
      id-pe-aaControls             OBJECT IDENTIFIER ::= { id-pe 6 }
      id-pe-ac-proxying            OBJECT IDENTIFIER ::= { id-pe 7 }

      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 }

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

        AttributeCertificateInfo ::= SEQUENCE {
          version        AttCertVersion DEFAULT v1,
          holder         Holder,
          issuer         AttCertIssuer,
          signature      AlgorithmIdentifier,
```



```
          serialNumber   CertificateSerialNumber,
          attrCertValidityPeriod   AttCertValidityPeriod,
          attributes     SEQUENCE OF Attribute,
          issuerUniqueID UniqueIdentifier OPTIONAL,
          extensions     Extensions     OPTIONAL
        }

        AttCertVersion ::= INTEGER {v1(0), 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
                       -- if present, version must be v2
        }

        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 {
  oldForm        GeneralNames,
  newForm    [0] SEQUENCE {
  issuerName               GeneralNames      OPTIONAL,
  baseCertificateId   [0]  IssuerSerial      OPTIONAL,
  objectDigestInfo    [1]  ObjectDigestInfo  OPTIONAL
   -- at least one of issuerName, baseCertificateId or --
   -- objectDigestInfo must be present --
   -- if newForm is used, version must be v2--
}

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,
     targetCertificate   [2] IssuerSerial,
     targetDigest        [3] ObjectDigestInfo
}

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
  }

  Clearance  ::=  SEQUENCE {
       policyId  OBJECT IDENTIFIER,
       classList ClassList DEFAULT {unclassified},
       securityCategories
                 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
```

Appendix C: Changes History

<<This Appendix to be deleted after last call>>

This appendix lists major changes since the previous revision.

Major changes since last revision:

Changes from -01 to -02

1. Re-Synchronized with X.509 DAM
2. Deleted AC chains concept
3. Moved AAControls to "optional features" section
4. Samples will be a separate draft
5. Revocation: now using X.509 DAM (noRevAvail) and standard 2459 mechanisms only
6. Deleted the special wildcard target "ALL"

Changes from -00 to -01

1. Re-structured conformance to profile + options as per Oslo consensus
2. Moved acquisition protocol (LAAP)_to separate I-D
3. Removed restrictions entirely
4. Added new AC revocation options
5. Added optional support for use of objectDigestInfo for keys
6. Added optional support for chains of ACs
7. Changed some syntax:
   Added UTF8String to IetfAttrSyntax value choice
   Split target & proxy extensions, removed owner from proxyInfo
8. Allocated PKIX OIDs (note: check with repository before using these, the PKIX arc is currently available at http://www.imc.org/ietf-pkix/pkix-oid.asn)
9. Added compiled ASN.1 module