INTERNET-DRAFT
PKIX Working Group
expires in six months

S. Farrell
SSE
R. Housley
Spryus
April 1999

An Internet AttributeCertificate Profile for Authorization

<draft-ietf-pkix-ac509prof-00.txt>

Status of this memo

<<Comments are contained in angle brackets like this.>>

Abstract

Authorization support is required for various Internet protocols, for example, TLS, CMS and their consumers, and others. The X.509 AttributeCertificate provides a structure that can form the basis for such services ([X.509], [XPDAM]). This specification defines two profiles (basic and proxiable) for the use of X.509 AttributeCertificates to provide such authorization services.

1. Introduction

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

The provision of authentication, data integrity and confidentiality services for current Internet protocols is well understood and many secure transports are defined (e.g. TLS, IPSEC, etc.). In many applications these services are not sufficient (or too cumbersome to administer) to provide the type of authorization services required.

[RFC2459] specifies a profile for the use of X.509 public key certificates in Internet protocols. This type of certificate is typically used as an "identity" certificate, that is, it contains a certified name and public key, and any entity that can use the corresponding private key is treated as the named entity.

When considering authorization, one is often less interested in the identity of the entity than in some other attributes, (e.g. roles, account limits etc.), which should be used to make an authorization decision.

In many such cases, it is better to separate this information from the identity for management, security, interoperability or other reasons. However, this authorization information also needs to be protected in a fashion similar to a public key certificate - the name for the structure used is an attribute certificate (an AC) which is a digitally signed (certified) set of attributes.

An AC is a structure that is similar to an X.509 public key certificate [RFC2459] with the main difference being that it contains no public key. The AC typically contains group membership, role, clearance and other access control information associated with the AC owner. The base syntax for ACs is also defined in the X.509 standard (making the term X.509 certificate ambiguous!). This document specifies a profile of the X.509 AC suitable for authorization purposes in Internet protocols.

When making an access decision based on an AC, an access decision function may need to ensure that the appropriate AC owner 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 requestor and that PKC has been used to authenticate the request.

As there is often confusion about the difference between public key certificates (PKCs) and attribute certificates (ACs), an analogy may help. A PKC can be considered to be like a passport: it identifies the owner, tends to last for a long period and shouldn't be too easy to get. An AC is more like an entry visa in that it is typically issued by a different authority and doesn't last as long. As acquiring an entry visa typically requires presenting a passport, getting a visa can be a simpler process.

In conjunction with authentication services ACs provide a means to transport authorization information securely to applications. 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 domains 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 and 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.

The diagram below shows an abstract view of the exchanges that may involve ACs. This profile does not specify protocol for all of these exchanges, though a limited case of client and server acquisition is defined below.

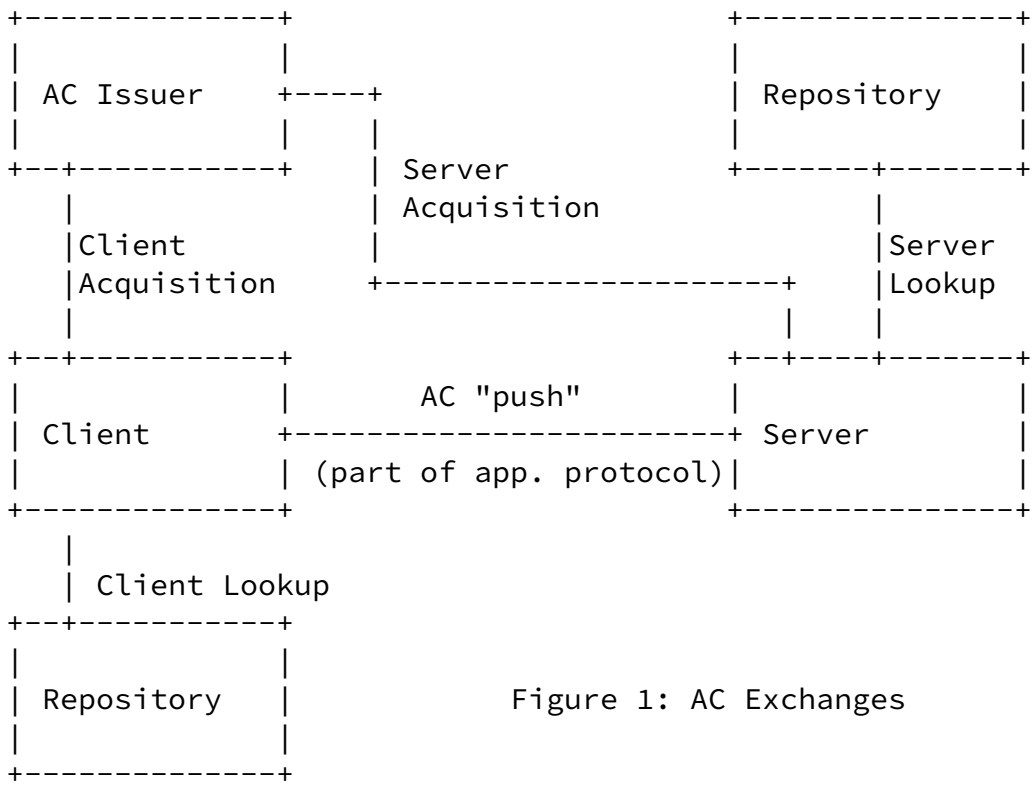

Figure 1: AC Exchanges

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 a limited AC acquisition protocol
Section 8 contains a conformance statement

Appendices contain samples, a "compilable" ASN.1 module for this specification and a list of open issues.

2. Terminology

Term	Meaning
AC	AttributeCertificate
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

AC owner	the entity indicated (perhaps indirectly) in the subject field of the AC
Client	the entity which is requesting the action for which authorization checks are to be made
LAAP	Limited AC Acquisition Protocol
LRP	LAAP responder
LRQ	LAAP requestor
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

The following are the requirements that the "full" profile defined here meets.

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 scheme.

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 SSE and the "Administrators group" as defined by Widgets inc.
5. ACs should support the encryption of some, or all, attributes (e.g. passwords for legacy applications). It should be possible for such an encrypted attribute to be

deciphered by an appropriate AC verifier even where the AC has not been received directly from the AC owner (i.e. where the AC is proxied).

Targeting of ACs:

6. It should be possible to "target" an AC. This means that a given AC may be "targeted" at one, or a number of, servers/services in the sense that a trustworthy non-target will reject the AC for authorization decisions.

Proxying:

7. It should be possible for a server to proxy an AC when it acts as a client (for another server) on behalf of the AC owner.
8. Proxying should 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.
9. Support for chains of proxies (with more than one intermediate server) is required.

Push vs. Pull

10. ACs should be defined so that they can either be "pushed" by the client to the server, or "pulled" by the server from a network service (whether the AC issuer or

an online repository).

This profile specifically imposes no requirements for:

1. The meaning of a chain of ACs
2. AC translation

Support for such features may be part of some other profile.

4. The AC Profile

This section specifies the profile of the X.509 AC which is to be supported by conforming implementations.

4.1 X.509 AttributeCertificate Definition

X.509 contains the definition of an AttributeCertificate given below. Types that are not defined can be found in [RFC2459].

<<This definition is from the PDAM.>>

```
AttributeCertificate ::= SIGNED {
     acinfo         AttributeCertificateInfo
}

AttributeCertificateInfo ::= SEQUENCE {
     version   AttCertVersion DEFAULT v1,
     owner     CHOICE{
          baseCertificateID   [0] IssuerSerial,
               -- the issuer and serial number of
```

```
                    -- the owner's Public Key Certificate
               entityName          [1] GeneralNames,
                    -- the name of the claimant or role
               objectDigestInfo    [2] ObjectDigestInfo
                    -- if present, version must be v2
          },
          issuer              CHOICE {
               baseCertificateId   [0]  IssuerSerial,
               issuerName          [1]  GeneralNames
          },
          --AA that issued the attribute certificate
          signature      AlgorithmIdentifier,
          serialNumber   CertificateSerialNumber,
          attrCertValidityPeriod   AttCertValidityPeriod
          attributes     SEQUENCE OF Attribute,
          issuerUniqueID UniqueIdentifier OPTIONAL,
          extensions     Extensions     OPTIONAL
     }

     AttCertVersion ::= INTEGER {v1(0), v2(1) }

     ObjectDigestInfo ::= SEQUENCE {
          digestAlgorithm     AlgorithmIdentifier,
          objectDigest        OCTET STRING
     }

     IssuerSerial  ::=  SEQUENCE {
          issuer         GeneralNames,
```



```
          serial         CertificateSerialNumber,
          issuerUID      UniqueIdentifier OPTIONAL
     }

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

4.2 Object Identifiers

The following OIDs are used:

```
<<
for interop testing purposes the SSE OID sse-ac-tst may
be used instead of ietf-ac.
sse-id         OBJECT IDENTIFIER ::= { 1 3 6 1 4 1 1201 }
```

```
sse-ac-tst     OBJECT IDENTIFIER ::= { sse-id 56 }
>>

ietf-ac              OBJECT IDENTIFIER ::= <<tbs>>
ietf-ac-extensions   OBJECT IDENTIFIER ::= { ietf-ac 1}
ietf-ac-attributes   OBJECT IDENTIFIER ::= { ietf-ac 2}
```

4.3 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).

4.3.1 version

This must be the default value of v1, i.e. not present in encoding.

<<If we allow objectDigest then the above will have to change. There's also a comment to the PDAM which may cause a change here.>>

4.3.2 owner

For any protocol 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 owner field MUST use the baseCertificateID.

With the baseCertificateID option, the owner's PKC serialNumber and issuer MUST be identical to the AC owner

field. The PKC issuer MUST have a non-NULL X.500 name which is to be present as the single value of the of the owner.issuerSerial.issuer construct in the directoryName field. The owner.issuerSerial.issuerUID field MUST only be used if the owner's PKC contains an issuerUniqueID field.

The above means that the baseCertificateID is only usable with PKC profiles (like RFC2459) which mandate that the PKC issuer field contain a value.

If the owner 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, or, if the PKC subject is a "NULL" DN, then the entityName field MUST be identical to one of the values of the PKC subjectAltName field extension. Note that [RFC2459] mandates that the subjectAltNames extension be present if the PKC subject is a "NULL" DN.

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

AC's conforming to this profile MUST NOT use the objectDigest field.

<<Uses of objectDigest are for further study.>>

Any protocol conforming to this profile SHOULD specify which AC subject option is to be used and how this fits with e.g. peer-entity authentication in the protocol.

4.3.3 issuer

ACs conforming to this profile MUST use the issuerName choice which MUST contain one and only one GeneralName which MUST contain its non-null value in the directoryName field. This means that all AC issuers MUST have non-NULL X.500 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.

4.3.4 signature

Contains the algorithm identifier used to validate the AC signature.

This MUST be one of the following algorithms defined in [RFC2459] section 7.2: md5WithRSAEncryption, id-dsa-with-sha1 or sha-1WithRSAEncryption.

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

4.3.5 serialNumber

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 topmost bit in the DER encoding of the INTEGER value MUST NOT be a `1'B - this is to 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 a serialNumber.

Given the uniqueness and timing requirements above serial numbers can be expected to contain long integers, i.e. AC users MUST be able to handle more than 32 bit integers here.

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.

4.3.6 attrCertValidityPeriod

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

GeneralizedTime encoding is restricted as specified in [RFC2459] for the corresponding fields in a PKC.

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.

4.3.7 attributes

The attributes field gives information about the AC owner. When the AC is used for authorization this will

often contain a set of privileges. However, authorization

may also require support for "restrictions" - these are not carried within the attributes field (though they "belong" to the AC owner) but in the extensions field.

The attributes field contains a SEQUENCE OF Attribute. For a given AC each attribute type in the sequence MUST be unique, that is, only one instance of each attribute type can occur in a single AC. Each instance can however, be multi-valued.

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

Note that a conforming AC MAY contain an empty SEQUENCE, that is, no attributes at all.

Some standard attribute types are defined in section 4.5.

4.3.8 issuerUniqueID

This field MUST NOT be used.

4.3.9 extensions

The extensions field generally gives information about the AC as opposed to information about the AC owner. The exception is where restrictions are to be supported. If one regards a restriction as a qualification on a privilege then it is clear that restrictions must be implemented as a critical extension.

Section 4.4 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.

4.4 Extensions.

4.4.1 Restrictions

<<The authors solicit comment on whether support for restrictions is needed. The benefit is that they may allow the "positive" privilege syntax to be standardised more widely under the assumption that the corresponding restriction syntax need only be understood "locally". On the other hand, we can omit this entirely if not many people see any benefit.>>

A restriction is a "negative" privilege, for example an AC may "state" that the AC owner is a member of the administrative group except for purposes of backup. Restrictions would more properly be implemented as a separate field of the AC, but with the current syntax can only be supported via the use of a critical extension.

The value of this extension will be a SEQUENCE OF Attribute. The rule stated above for the AC attributes field (only one instance of each type etc.) applies here also.

Each restriction MUST correspond to one attribute present in the attributes field and must use the same attrType OID as the related attribute.

name	ietf-ac-restrictions
OID	{ ietf-ac-extensions 1 }
syntax	SEQUENCE OF Attribute
criticality	MUST be TRUE

4.4.2 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 owner 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 owner'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 track the behaviour of an AC owner without being able to identify the AC owner.

The server/service administrator in combination with the AC issuer MUST be able to identify the AC owner in cases where mis-behaviour 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 owner.

Of course, auditing could be based on the AC

issuer/serial pair, however, this method doesn't allow tracking the same AC owner 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 owner's PKC issuer/serial however, this will often allow the server/service administrator identify the AC owner.

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 owner 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".

name	ietf-ac-auditId
OID	{ ietf-ac-extensions 3 }
syntax	OCTET STRING
criticality	must be TRUE

4.4.3 AC Targeting and Proxying

In order to allow that an AC is "targeted" and to control proxying, the proxy 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 proxiable. Any server which receives the AC such that the owner and the authenticated peer-entity do not match MUST reject the AC.

When this extension is present we are essentially checking 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 recipient.

The targeting information consists of the direct

information (targets field) and an optional set of proxy information (proxies field). If the "direct check" or any of the "proxy" checks (see below) pass then the "targeting check" as a whole is successful.

The effect is that the AC owner can send 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 {
     owner     CHOICE {
          baseCertificateID   [0]  IssuerSerial,
          subjectName         [1]  GeneralNames,
          objectDigestInfo    [2]  ObjectDigestInfo
     },
     targets   [0]  Targets             OPTIONAL,
     proxies   [1]  SEQUENCE OF Targets OPTIONAL
}
Targets ::= SEQUENCE OF Target
Target ::= CHOICE {
     targetName          [0] GeneralName,
     targetGroup         [1] GeneralName
}
```

Where no proxies or targets are present then the entire field MUST be omitted, that is, a zero-length sequence of Targets MUST NOT be present. There MUST be at least one target or one proxy present, that is, one of the targets or proxies fields MUST be present.

We represent a special target, called "ALL" which is a wildcard as a targetName with the registeredID choice and a value of {ietf-ac-extensions 4 1}. This is an exception to the general rule stated above about the use of GeneralName choices.

The direct check passes if:

- the identity of the client as established by the underlying authentication service matches the owner field

```
        and
        (
            the targets field contains one targetName which
            is the "ALL" value
             or
             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.
        )
```

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 or it the targetGroup were "PRINTERS" and the AC verifier "knows" that it's a printer or print server.

A proxy check succeeds if

```
        (
             the identity of the sender as established by
             the underlying authentication service matches
             the owner field
             and
             (
                  the current server "matches" any one of
                  the proxy sets (where "matches" is as for
                  the direct check above)
             )
        )
        or
        (
             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 owner. 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	ietf-ac-targeting
OID	{ ietf-ac-extensions 4 }
syntax	ProxyInfo
criticality	must be TRUE

4.4.4 authorityKeyIdentifier

The authorityKeyIdentifier extension as profiled in [RFC2459] MAY be used to assist the AC verifier in checking the signature of the AC. The [RFC2459] description should be read as if "CA" meant "AC issuer". As with PKCs this extension SHOULD be included in ACs.

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

4.4.5 authorityInformationAccess

The authorityInformationAccess extension as profiled in [RFC2459] MAY be used to assist the AC verifier in checking the revocation status of the AC. See section 6 on revocation below for details.

The following accessMethod is used to indicate that revocation status checking is not provided for this AC:

```
ietf-ac-norevstat   OBJECT IDENTIFIER ::=
                         { ietf-ac-extensions 5}
```

The accessLocation field MUST contain a NULL directoryName.

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

4.5 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 organisation) allocates attribute values, but where multiple AC issuers are deployed for performance, network or other reasons.
2. It allows the type of the attribute (privilege, restriction) to be made explicit which helps server implementations that provide an API on top of an AC validation module.

3. The syntaxes allowed for values are restricted to OCTET STRING and OID, which reduces some of the matching complexities associated with GeneralName.

<<The authors solicit comment on whether this flexibility is required. The alternative would be to encourage the use of attributes which have a GeneralName syntax and to mandate this for role/group etc..>>

```
IetfAttrSyntax ::= SEQUENCE OF {
     type      INTEGER {
                         privilege(0),
                         restriction(1),
                         other(2)
               }
                DEFAULT privilege,
```

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

4.5.1 Service Authentication Info

This attribute type identifies the AC owner to the server/service by a name and with optional authentication information. Typically this will contain a username/password pair for a "legacy" application (and hence MAY need to be encrypted).

This attribute type will typically be encrypted if the authInfo field contains sensitive information (e.g. a password).

```
name      ietf-ac-authInfo
OID       { ietf-ac-attributes 1}
Syntax    SvceAuthInfo
values:   Multiple allowed

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


4.5.2 Access Identity

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

```
name      ietf-ac-accessId
OID       { ietf-ac-attributes 2}
syntax    SvceAuthInfo
values:   Multiple allowed
```

4.5.3 Charging Identity

This attribute type identifies the AC owner for charging purposes.

```
name      ietf-ac-chargingId
OID       { ietf-ac-attributes 3}
syntax    IetfAttrSyntax
values:   Multiple allowed
```

4.5.4 Group

This attribute carries information about group memberships of the AC owner.

<<Might it be more useful to define OS-specific group attribute types which map to UNIX gids and/or NT SIDs? Even with that, application defined groups will be needed - should they use a standard group attribute or should appX-group attribute types be defined for each?>>

```
name      ietf-ac-group
OID       { ietf-ac-attributes 4}
syntax    IetfAttrSyntax
values:   Multiple allowed
```

4.5.5 Role

This attribute carries information about role allocations of the AC owner.

```
name      ietf-ac-role
OID       { ietf-ac-attributes 5}
syntax    IetfAttrSyntax
values:   Multiple allowed
```


4.5.6 Clearance

This attribute carries clearance (security labelling) information about the AC owner.

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

     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 SECURITY-CATEGORY,
          value     [1]  ANY DEFINED BY type
     }

     SECURITY-CATEGORY MACRO  ::=
     BEGIN
     TYPE NOTATION ::= type | empty
     VALUE NOTATION ::= value (VALUE OBJECT IDENTIFIER)
     END
```

4.5.7 EncryptedAttributes

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.

Only one encryptedAttributes 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 would be to merge attributes following decryption in order to re-establish the "once only" constraint.

```
name      ietf-ac-encAttrs
OID       { ietf-ac-attributes 6}
Syntax    ContentInfo
values    Multiple Allowed
```

4.6 PKC Extensions

Public key certificate extensions which assist in AC handling are defined in this section.
<<just one for now, and hopefully, always!>>

4.6.1 AAControls

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. The use of AAControls is further described in section 5.

```
aaControls EXTENSION ::= {
   SYNTAX         AAControls
   IDENTIFIED BY  { ietf-ac-pkcexts-aaControls}
}
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 [RFC2459], 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.

5. AttributeCertificate 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 PKC MUST be verified in accordance with [RFC2459]
2. if the AC issuer is not directly trusted as an AC issuer (by configuration or otherwise), then the AC issuer's certification path must satisfy the additional PKC checks described below
3. the time of evaluation 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)

4. 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
5. the AC targeting check MUST pass (see section 4.4.3 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.

The following additional certification path checks (referred to in (2) above) MUST all succeed:

1. some CA on the AC's certificate path MUST be directly trusted to issue PKCs which preceed 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 ietf-ac-encAttrs type MUST also be checked)

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 do not require revocation support. However, long-lived ACs and environments where ACs enable high value transactions MAY require revocation support.

In such cases, AC revocation status MAY be checked using the methods described in [RFC2459], but substituting the AC issuer wherever a CA is mentioned.

Note however that this does not impose a requirement for conformant AC issuers to be able to issue CRLs.

Where an AC issuer does not support revocation status checks for a particular AC, then an authority information access extension (id-pe-authorityInfoAccess) MUST be present and critical in the AC to indicate this. Where no authority information access is present, then the AC issuer is implicitly stating that revocation checks are supported and mechanisms in accordance with [RFC2459] MUST be provided to allow AC verifiers to establish the revocation status of the AC.

The accessMethod used to handle this case is described above.

7. Limited AC Acquisition Protocol

<<Note that this section is very likely to change and may be removed, in particular if it is found that CMP or CMC can be suitably extended to support AC acquisition. If the WG reaches a consensus that a new protocol is needed then this section may move to a separate I-D. Even with a new protocol, it would be appropriate to examine extending CMP/CMC to handle more general AC management tasks. Basically, this is a "strawman">>

There is clearly a requirement for an AC management protocol (or protocols, like [CMP] and [CMC]). Such management protocols are not specified in this document. There is also a requirement for a specification of an LDAP schema, which is also not specified here.

In addition to such protocols, which may be more suited to management of long-term or more sensitive (i.e. more "powerful") ACs, there is a requirement for a very

simple, explicitly limited AC acquisition protocol.

This protocol is required for cases where an AC user wishes to acquire a "current" AC for an entity (possibly itself) leaving almost all details as to the content of the AC to the AA or whatever network service acts on its behalf.

We call this protocol the Limited AC Acquisition Protocol (LAAP). The are two entities involved, the LAAP requestor (LRQ) and LAAP responder (LRP). The LR is typically an AC owner or an AC verifier; the LRP is typically the AA itself.

LAAP is designed as a single-shot request/response protocol with no polling, retries, etc.

The one and only feature of this protocol is to request an AC for a particular entity that may be either the requestor or some other entity. The response is the requested AC or an error.

The security of the request and response (e.g. whether the requestor is authenticated or not) is out of scope and a matter for LAAP implementers. For example, an LRP may be configured so that it only ever issues ACs if the request is received over an authenticated channel (e.g. TLS with client authentication), or it may only issue "guest" privileges when the LRQ is not the owner of the AC.

The protocol consists of a request message that may specify the identity of the AC owner (for the third party case), with an optional "profile". A profile is to be interpreted as a bilaterally agreed string that is mapped to a set of AC contents by the LRP.

```
LACRequestMessage ::= SEQUENCE {
          owner     [0] CHOICE{
               baseCertificateID   [0] IssuerSerial,
                    -- the issuer and serial number of
                    -- the owner's Public Key Certificate
               entityName          [1] GeneralNames,
                    -- the name of the claimant or role
               objectDigestInfo    [2] ObjectDigestInfo
```

```
                    -- if present, version must be v2
          } OPTIONAL,
          profile   [1] UTF8String OPTIONAL
   }
```

<<Note that this message syntax omits any "proof" that the LRQ has some valid reason to ask for an AC for the owner. It would be (at least) nice to be able to include such "proof", but can't be specified here since it might depend on the client/server authentication. Other than via the profile field, the LRQ also cannot specify the target(s) where the AC will have to be verified. We need to consider if these are needed or not.>>

Each field is described below.

"owner": when present this specifies that the LRQ wishes to acquire an AC for this owner. When absent, it means that the LRQ is requesting an AC for itself (the LRP should use the identity established from whatever underlying authentication is available). The rules for the owner field in the AC apply here (e.g. no use of objectDigestInfo).

"profile": when present this is an request to the LRP that an AC matching a certain profile be returned. The definition of profiles is not in scope for this specification and is expected to be a local matter. This field allows some simple switching.

Note that this definition means that the minimal LAAP request message consists of the octets `3000'H, an empty sequence. This message means "give me my current default AC please".


```
   LACResponseMessage ::= CHOICE {
        ac        [0]  AttributeCertificate,
        errorInfo [1]  ErrorMsgContent -- from [CMP]
   }
```

When an LRQ receives an AC from an LRP it SHOULD verify the AC. In addition the LRQ SHOULD ensure that the AC "matches" the LAAP request issued. The only matching

which applies in general is to ensure that the LAAP request owner field and the AC owner field are identical. Implementations may of course include additional checks.

We define the following HTTP based transport for LAAP.

An LRQ should send a HTTP POST request to the LRP, the POST data should consist of the DER encoding of the LACRequestMessage. The response is expected to have the MIME type "application/x-laapmsg" with the message data containing the DER encoding of the LACResponseMessage.

<<how an LRQ knows the URL for an LRP is TBS>>

8. Conformance

This specification defines two levels of conformance, basic and proxy-enabled. For each level the actors involved must meet different requirements. The intention is that support for basic conformance should allow for freely interoperable but fairly inflexible and "featureless" AC based authorization. Proxy-enabled conformance requires more effort from implementers, may not be as widely interoperable and is harder to administer, but does offer much more flexibility and many more features.

A proxy-enabled AC issuer MUST be able to produce all of the attribute types and extensions specified above.

A proxy-enabled AC verifier MUST "support" all of the attribute types and extensions specified above.

"Support" in the previous paragraph means more than just parsing. It means that the AC verifier MUST be able to reject any AC which should not be valid at that target and MUST be able to make any attributes and extensions which were not fully processed available to the calling application.

A proxy-enabled AC issuer is responsible to ensure that no AC produced could be accepted by a basic AC verifier in such a way as to cause a security breach.
<<dunno if that can happen but it needs to be checked>>

Basic conformance for an AC issuer means support for production of ACs which:

1. MUST use the baseCertificateID owner field alternative
2. MUST NOT be post-dated
3. MAY contain AccessIdentity, Group and/or Role attributes with multiple values
4. MUST NOT contain any other attributes which cannot safely be ignored by an AC verifier
5. MAY contain the AuthorityKeyIdentifier extension
6. MUST contain no critical extensions (and hence is not proxiable) except for authorityInformationAccess where revocation status checks are not provided
7. MUST NOT contain encrypted attributes

Basic conformance also requires support for the AAControls PKC extension. A basic AC issuer MUST also support LAAP as specified in section 7 above.

Basic conformance for an AC verifier means support for the validation of ACs which are produced by basic AC issuers.

A basic AC verifier MAY ignore the presence of any unsupported attributes or extensions (of course it must reject all ACs which contain unsupported critical extensions) and need only make the values of the remaining attributes available to applications.

A basic AC verifier MUST support the AAControls PKC extension.

9. Security Considerations

<<tbs>>

10. References

[CMC] Myers, M., et al. "Certificate Management Messages over CMS", draft-ietf-pkix-cmc-03.txt, March 1999.

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

[CMS] Housley, R., "Cryptographic Message Syntax", draft-ietf-smime-cms-12.txt, March 1999.

[ESS] Hoffman, P., "Enhanced Security Services for S/MIME", draft-ietf-smime-ess-12.txt, March 1999.

[RFC2459] Housley, R., Ford, W., Polk, T, & Solo, D., "Internet Public Key Infrastructure - X.509 Certificate and CRL profile", RFC2459.

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

[X.509] ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997.

[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.
[XPDAM] ISO 9594-8 Information Technology - Open systems Interconnection - The Directory: Authentication Framework - Proposed Draft Amendment 1: Certificate Extensions, September 1998.

Author's Addresses

Stephen Farrell,
SSE Ltd.
Fitzwilliam Court,
Leeson Close,
Dublin 2,
IRELAND

tel: +353-1-216-2910
email: stephen.farrell@sse.ie

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

email: housley@spyrus.com

Full Copyright Statement

Appendix B may be used in whole or in part without inclusion of the copyright notice. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process shall be

followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns. This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Appendix A: Samples

tbs

Appendix B: "Compilable" ASN.1 Module

<<tbs - will be supplied in '88 format>>

Appendix C: Open Issues

This appendix lists open issues to be resolved (in order of occurrence in the body of the document).

1. Additional introductory text is required to motivate the use of ACs
2. The ASN.1 has to be synched with the ISO FPDAM where necessary
3. An OID for ietf-ac has to be allocated
4. The objectDigest choice for owner has to be a MUST NOT or else profiled (and explained!)
5. Are "restrictions" needed?
6. Is "IetfAttrSyntax" needed?

7. Should OS-specific group attribute types be defined?
8. More explanatory text for encryptedAttributes is needed.
9. Is a new AC acquisition protocol required? If not, how are ACs acquired? If so, should it be part of this specification?
10. Are different conformance levels needed? If so, are these the right ones?
11. Security considerations text needed
12. References section to be fixed
13. Compilable ASN.1 and samples are needed