

PKIX Working Group
INTERNET-DRAFT
Expires in six months

S. Farrell
Baltimore Technologies
R. Housley
SPYRUS
October 1999

An Internet Attribute Certificate
Profile for Authorization
<[draft-ietf-pkix-ac509prof-01.txt](#)>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of \[RFC2026\]](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

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

Abstract

Authorization services are required for numerous Internet protocols, including TLS, IPSec, and S/MIME. The X.509 Attribute Certificate provides a structure that can form the basis for such services [X.509]. This specification defines a profile for the use of X.509 Attribute Certificates to provide authorization services for Internet protocols. Some optional features are also specified which are not required for conformance to the base profile.

Table of Contents

Status of this Memo.....	1
Abstract.....	1
Table of Contents.....	1
1 . Introduction.....	3

2.	Terminology.....	5
3.	Requirements.....	6
4.	The AC Profile.....	7

4.1	X.509 Attribute Certificate Definition.....	7
4.2	Object Identifiers.....	8
4.3	Profile of Standard Fields.....	9
4.3.1	version.....	9
4.3.2	owner.....	10
4.3.3	issuer.....	10
4.3.4	signature.....	10
4.3.5	serialNumber.....	11
4.3.6	attrCertValidityPeriod.....	11
4.3.7	attributes.....	12
4.3.8	issuerUniqueID.....	12
4.3.9	extensions.....	12
4.4	Extensions.....	12
4.4.1	Audit Identity.....	12
4.4.2	AC Targeting.....	13
4.4.3	authorityKeyIdentifier.....	14
4.4.4	authorityInformationAccess.....	14
4.4.5	crlDistributionPoints.....	15
4.5	Attribute Types.....	15
4.5.1	Service Authentication Info.....	16
4.5.2	Access Identity.....	16
4.5.3	Charging Identity.....	16
4.5.4	Group.....	17
4.5.5	Role.....	17
4.5.6	Clearance.....	17
4.6	PKC Extensions.....	18
4.6.1	AAControls.....	18
4.7	Profile of AC Issuer's PKC.....	19
5.	Attribute Certificate Validation.....	19
6.	Revocation.....	21
6.1.1	"Never revoke" method.....	21
6.1.2	"Pointer from above" method.....	22
6.1.3	"Pointer in AC" method.....	22
7.	Optional Features.....	22
7.1	Attribute Encryption.....	22
7.2	Proxying.....	23
7.3	Use of ObjectDigestInfo.....	25
7.4	AC Chaining.....	26

8. Security Considerations.....	27
9. References.....	27
Author's Addresses.....	28
Full Copyright Statement.....	28
Appendix A: "Compilable" ASN.1 Module.....	29
Appendix B: Samples.....	32
Appendix C: Changes this version / Open Issues.....	32

INTERNET-DRAFT

October 1999

[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) [\[X.509\]](#), [\[RFC2459\]](#) 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 owner. 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 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 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 owner, 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.

In conjunction with authentication services, ACs provide a means to securely provide authorization information to applications. However, there are a number of possible communication paths that an AC may

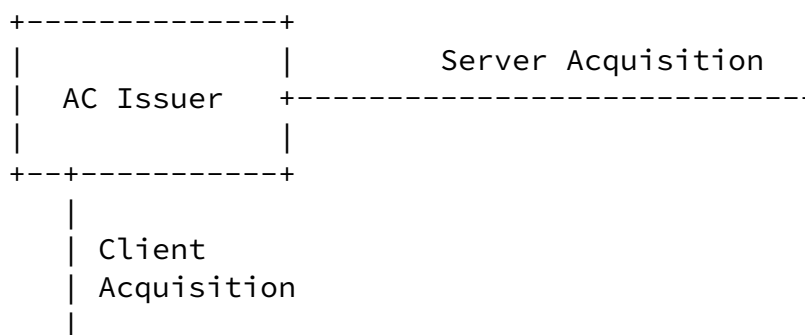
take.

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 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.



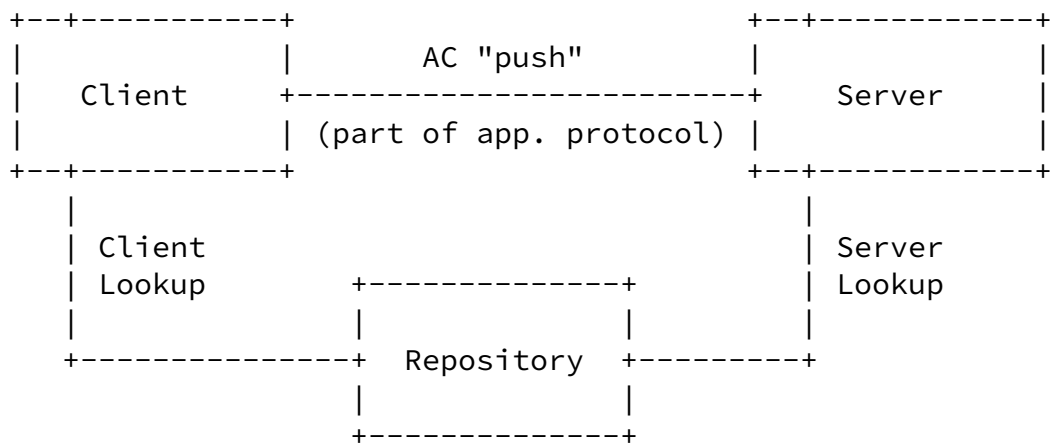


Figure 1: AC Exchanges

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

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 owner	the entity indicated (perhaps indirectly) in the owner 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

This section presents a profile for attribute certificates that will foster interoperability. This section is based upon the X.509 attribute certificate format defined in [X.509]. The ISO/IEC/ITU documents use the 1993 version of ASN.1; while this document uses the 1988 ASN.1 syntax, the encoded certificate and standard extensions are equivalent. This section also defines private extensions for the Internet community.

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.

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 [[RFC2459](#)].

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

```
AttributeCertificateInfo ::= SEQUENCE {
    version             AttCertVersion DEFAULT v1,
```


owner	Owner,
issuer	AttCertIssuer,
signature	AlgorithmIdentifier,
serialNumber	CertificateSerialNumber,
attrCertValidityPeriod	AttCertValidityPeriod
attributes	SEQUENCE OF Attribute,
issuerUniqueID	UniqueIdentifier OPTIONAL,

```

        extensions      Extensions      OPTIONAL
    }

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

Owner ::= SEQUENCE {
    baseCertificateID    [0] IssuerSerial OPTIONAL,
        -- the issuer and serial number of
        -- the owner'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 {
    digestAlgorithm      AlgorithmIdentifier,
    objectDigest         OCTET STRING
}

AttCertIssuer ::= SEQUENCE {
    issuerName           GeneralNames OPTIONAL,
    baseCertificateId    [0] IssuerSerial OPTIONAL
}

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

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

```

[4.2](#) Object Identifiers

This section 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.

The following OIDs are imported from [\[RFC2459\]](#):

```
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 registeredID form of name for targets and proxies is defined (see [section 4.4.2](#) below):

```
id-pe-ac-targeting-all    OBJECT IDENTIFIER ::=
    { id-pe-ac-targeting 1 }
```

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-role                OBJECT IDENTIFIER ::= { id-aca 5 }
```


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.

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 owner 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`, 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.

[4.3.5](#) 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.

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 [[RFC2459](#)], [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.

[4.3.7](#) 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.

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 users 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. <<Note: This is no longer required since we've dropped support for restrictions, so it will disappear in the next revision unless there's an explicit consensus for keeping it.>>

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.

[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](#) 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 behavior 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 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 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	id-pe-ac-auditIdentity
OID	{ id-pe 4 }
syntax	OCTET STRING
criticality	must be TRUE

[4.4.2](#) 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.

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
}
```

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

The targets check passes if:

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

```
name          id-pe-ac-targeting
OID           { id-pe 5 }
```


syntax	Targets
criticality	must be TRUE

[4.4.3](#) 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

[4.4.4](#) 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:

```
id-ad-noRevStat OBJECT IDENTIFIER ::=
    { id-ad 3 }
```

The following accessMethod is used to indicate that revocation status checking is provided for this AC, using the OCSP protocol defined in [[RFC2560](#)]:

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

The following accessMethod is used to indicate that revocation status checking is provided "below" this PKC or AC:

```
id-ad-acRevStatusLocation OBJECT IDENTIFIER ::=
    { id-ad 4 }
```

The accessLocation field MUST contain a NULL directoryName.

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

[4.4.5](#) crlDistributionPoints

The crlDistributionPoints 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.

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

[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 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 and OID, which reduces some of the matching complexities associated with GeneralName. 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 OF SEQUENCE {  
    policyAuthority[0] GeneralNames    OPTIONAL,  
    values          SEQUENCE OF CHOICE {  
                        octets          OCTET STRING,
```

```

                                oid      OBJECT IDENTIFIER,
                                string    UTF8String
                                }
                                }

```

[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      id-aca-authenticationInfo
OID       { id-aca 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      id-aca-accessIdentity
OID       { id-aca 2 }
syntax    SvceAuthInfo
values:   Multiple allowed

```

[4.5.3](#) Charging Identity

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

```

name      id-aca-chargingIdentity
OID       { id-aca 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 id-aca-group
 OID { id-aca 4 }
 syntax IetfAttrSyntax
 values: Multiple allowed

[4.5.5](#) Role

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

name id-aca-role
 OID { id-aca 5 }
 syntax IetfAttrSyntax
 values: Multiple allowed

[4.5.6](#) Clearance

This attribute (imported from [X.501]) carries clearance (security labeling) 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.501]
 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)

INTERNET-DRAFT

October 1999

```

    }

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

    -- original syntax with MACRO
    -- <<is the above equivalent??>>
    -- 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

```

[4.6](#) PKC Extensions

Public key certificate extensions which assist in AC handling are defined in this section. At the moment only one new extension is defined.

[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](#).

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

```

aaControls EXTENSION ::= {
    SYNTAX          AACControls
    IDENTIFIED BY   { id-pe-aaControls }
}
AACControls ::= 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 AACControls 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.

[4.7](#) Profile of AC Issuer's PKC

The AC Issuer's PKC MUST conform to [[RFC2459](#)] and 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.

If the AC issuer supports revocation of ACs then the AC issuer's PKC SHOULD contain an authorityInfoAccess extension with a new accessMethod which assists the AC verifier in checking the status of an AC.

The new accessMethod is:

id-ad-acRevStatusLocation OBJECT IDENTIFIER ::= { id-ad 4 }

The accessLocation field MUST contain a single GeneralName containing either an X.500 Name or a URL. If accessLocation contains an X.500 Name, then this is the name of a directory entry where a revocation list for ACs issued by this AC issuer should be present as a value of the attributeCertificateRevocationList attribute. If accessLocation contains a URI, then this specifies the transport used for OCSP [[RFC2560](#)] requests. The AC issuer MUST, of course, maintain an OCSP responder at this location.

Note that in contrast to the use of authorityInfoAccess described in [section 4.4.4](#), in this case the extension is not present in the AC, but rather in the AC issuer's PKC.

[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 PKC MUST be verified in accordance with [[RFC2459](#)].
2. The AC issuer's PKC MUST also conform to the profile specified in [section 4.7](#) above.
3. 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
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 4.4.3](#) 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 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).

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

<<Input is solicited on the suitability of the 3-scheme approach.>>

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.

"Pointer from above" scheme: The PKC (or AC see [section 7.4](#)) of an AC issuer may "point" to sources of revocation status information for all ACs issued by that AC issuer, (with the exception of those marked using the never-revoke method above).

"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).

The never revoke scheme requires a new authorityInfoAccess accessMethod. The pointer from above scheme also requires a new authorityInfoAccess accessMethod. The pointer in AC scheme is as specified in [[RFC2459](#)] and [[RFC2560](#)].

The never revoke scheme MUST be supported, the other schemes SHOULD be supported.

[6.1.1](#) "Never revoke" method

Where an AC issuer does not support revocation status checks for a particular AC, then an authority information access extension (id-pe-authorityInfoAccess) with an id-ad-noRevStat accessMethod as specified in [section 4.4.4](#) above MUST be present and critical in the AC to indicate this.

Where no authority information access is present with this accessMethod, then the AC issuer is implicitly stating that revocation status checks are supported and one of the other methods below MUST be provided to allow AC verifiers to establish the revocation status of the AC.

[6.1.2](#) "Pointer from above" method

In this case the AC issuer's PKC contains an authority information access extension with an id-ad-acRevStatusLocation accessMethod as described in [section 4.7](#) above.

[6.1.3](#) "Pointer in AC" method

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

In these cases, the AC contains either an authorityInfoAccess or crlDistributionPoints extensions as defined in [[RFC2459](#)] and [[RFC2560](#)] respectively.

[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 {
```

Farrell & Housley

[Page 22]

INTERNET-DRAFT

October 1999

```
    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

Farrell & Housley

[Page 23]

INTERNET-DRAFT

October 1999

In some circumstances, a server needs to proxy an AC when it acts as a client (for another server) on behalf of the AC owner. Such proxying needs 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.

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 owner 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

A proxy check succeeds if

```
(
    the identity of the sender as established by
    the underlying authentication service matches
    the owner field of the AC
    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	id-pe-ac-proxying
OID	{ id-pe 7 }
syntax	ProxyInfo
criticality	must be TRUE

[7.3](#) Use of ObjectDigestInfo

<<In order to keep it simple, I've only allowed for a hash over a key, a hash over a certificate is thus not supported. If this or any other form of hash were allowed, then we'll need a digestedObjectInfo extension as well.>>

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 `owner` field allows support for this requirement.

If the owner 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 `owner` field of the AC. For example, this allows production of ACs that are linked to public keys rather than names or certificates, or production of ACs which contain privileges associated with an executable object (e.g. a Java class).

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 [\[RFC2459\]](#) and [\[ECDSA\]](#) for encoding keys MUST be followed.

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 \[RFC2459\]](#). 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 \[RFC2459\]](#) and those specified in [\[ECDSA\]](#).

[7.4](#) AC Chaining

[Section 5](#) above specifies a way of embedding AAControls into PKCs in order to control the attribute types for which an AA will be trusted by an AC verifier.

There are two drawbacks to this mechanism:

- PKC issuers have to know about authorization attribute types
- It is likely to require more frequent changes to AA's PKCs

These problems can be avoided by placing the equivalent information into an AC for which the owner is an AA. However, this mechanism requires chaining of ACs and thus imposes possibly significant costs both in terms of implementation and deployment complexity.

In order to use this feature, an AC verifier presented with an AC, (belonging say to an end entity, call this EE-AC), must retrieve an AC which is owned by the issuer of EE-AC (call this AA-AC). The AC verifier next verifies AA-AC, extracts the AAControls information from AA-AC and uses this to decide which attributes from EE-AC should be trusted.

Of course, the issuer of AA-AC may or may not be directly trusted by the AC verifier for the required attributes. In such a case, the AC verifier may have to retrieve another AC (AA2-AC), etc. until it finds one issued by a directly trusted AC issuer for each of the relevant attributes.

AC verifiers which support this feature MUST also support the use of aaControls placed within PKCs.

When verifying an AC, the verifier needs to determine when a chain of ACs is needed.

When AAControls are present in an AC, they are placed as an extension of the AC, using the same extension defined in [section 4.6.1](#) above.

When chaining ACs the following additional verification rules apply

1. EE-AC.issuer and AA-AC.owner MUST contain the same value
2. At the time of evaluation all ACs in the chain MUST be valid

<<probably needs more about the AC chain validation algorithm>>

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.

There is often a requirement to map between the authentication supplied by a particular protocol (e.g. TLS, S/MIME) and the AC owner'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 owner may be authenticated using other means. Implementers SHOULD be very careful in mapping the authenticated identity to the AC owner.

9. 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.
- [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-01.txt](#), June 1999.
- [[RFC2459](#)] Housley, R., Ford, W., Polk, T., & Solo, D., "Internet Public Key Infrastructure - X.509 Certificate and CRL profile", [RFC2459](#).
- [[RFC2560](#)] Myers, M., et al., " X.509 Internet Public Key Infrastructure - Online Certificate Status Protocol - OCSP", [RFC2560](#).
- [[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.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.

Farrell & Housley

[Page 27]

INTERNET-DRAFT

October 1999

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

Copyright (C) The Internet Society (date). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. In addition, the ASN.1 module presented in [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.

Farrell & Housley

[Page 28]

INTERNET-DRAFT

October 1999

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: "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-pe-ac-targeting-all    OBJECT IDENTIFIER ::=
                            { id-pe-ac-targeting 1 }

id-acca                    OBJECT IDENTIFIER ::= { id-pkix 10 }

id-acca-authenticationInfo OBJECT IDENTIFIER ::= { id-acca 1 }
id-acca-accessIdentity     OBJECT IDENTIFIER ::= { id-acca 2 }
id-acca-chargingIdentity   OBJECT IDENTIFIER ::= { id-acca 3 }
id-acca-group              OBJECT IDENTIFIER ::= { id-acca 4 }
id-acca-role               OBJECT IDENTIFIER ::= { id-acca 5 }

```

```

id-acca-encAttrs           OBJECT IDENTIFIER ::= { id-acca 6 }

id-ad-noRevStat            OBJECT IDENTIFIER ::= { id-ad 3 }
id-ad-acRevStatusLocation OBJECT IDENTIFIER ::= { id-ad 4 }

```

```

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

```

```

AttributeCertificateInfo ::= SEQUENCE {
    version      AttCertVersion DEFAULT v1,
    owner        Owner,
    issuer       AttCertIssuer,
    signature    AlgorithmIdentifier,
    serialNumber CertificateSerialNumber,
}

```

```

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

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

Owner ::= SEQUENCE {
    baseCertificateID  [0] IssuerSerial OPTIONAL,
        -- the issuer and serial number of
        -- the owner'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 {
    digestAlgorithm    AlgorithmIdentifier,
    objectDigest       OCTET STRING
}

AttCertIssuer ::= SEQUENCE {
    issuerName          GeneralNames OPTIONAL,
    baseCertificateId  [0] IssuerSerial OPTIONAL
}

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
    }

IetfAttrSyntax ::= SEQUENCE OF 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 B: Samples

<<TBS>>

Appendix C: Changes this version / Open Issues

This appendix lists major changes since the previous revision and open issues to be resolved (in order of occurrence in the body of the document).

Major changes since last revision:

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

Open issues remaining:

1. Should an AC without any attributes be allowed?
2. Should OS-specific group attribute types be defined?
3. Is the expansion of the SecurityCategory MACRO correct?
4. Are three revocation schemes needed? Correct?

5. Should more types of objectDigestInfo be allowed?
6. AC chain section needs more description of chain validation.
7. Samples - should they be a separate draft?