Transport Layer Security Working Group S. Farrell
INTERNET-DRAFT SSE
Expires Feb. 20, 1999. August 20, 1998

An Internet AttributeCertificate Profile for Authorization

<draft-ietf-tls-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 which can form the basis for such services [X.509]. This specification defines two profiles (a simple one and a "full" one) for the use of X.509 AttributeCertificates to provide such authorization services.

1. Introduction

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.

AttributeCertificates (ACs) provide a method of overcoming these problems. An AC is a structure which is similar to an X.509 public key certificate 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.

In conjunction with authentication services ACs provide the means to securely transport authorization information to applications.

The next section introduces some simple terminology. This is followed a brief statement of requirements, then by the definition of the "full" profile. The following section describes an algorithm for AC validation. Next a very limited, "simple" profile is defined and this is followed by a description of security considerations related to use of this specification. Appendices contain sample ACs and a compilable ASN.1 module.

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
PKC	Public Key Certificate - uses the type ASN.1 Certificate defined in X.509. 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 which 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 certificates. Short validity periods mean that ACs can be usable without mandating 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 delegated).

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.

Delegation:

7. It should be possible for a server to delegate an AC when it acts as a client (for another server) on behalf of the AC owner.
8. Delegation should be under the AC issuer's control, so that not every AC is delegatable and so that a given delegatable AC can be delegated in a targeted fashion.
9. Delegation should support chains of delegation where more than one intermediate server is used.

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 a directory service).

This profile specifically imposes no requirements for:

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

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

From this point in the document, the use of MUST, SHOULD etc. is in conformance to [RFC2119]

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 which are not defined can be found in [PKIX-1]

```
AttributeCertificate ::=
          SIGNED AttributeCertificateInfo

AttributeCertificateInfo ::= SEQUENCE {
     version        Version   DEFAULT v1,
     subject        CHOICE {
          baseCertificateID   [0]  IssuerSerial,
          subjectName         [1]  GeneralNames
     },
     issuer         GeneralNames,
     signature      AlgortihmIdentifier,
     serial         CertificateSerialNumber,
     validity       AttrCertValidityPeriod,
     attributes     SEQUENCE OF Attribute,
     issuerUID      UniqueIdentifier    OPTIONAL,
     extensions     Extensions          OPTIONAL
}
```

```
AttrCertValidityPeriod ::= SEQUENCE {
     notBefore      GeneralizedTime,
     notAfter       GeneralizedTime
}

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

4.2 Object Identifiers

The following OIDs are used:

```
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.

4.3.1 version

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

4.3.2 subject

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 subject field MUST use the baseCertificateID.

With the baseCertificateID option, the subject's PKC serialNumber and issuer MUST be identical to the AC subject 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 subject.issuerSerial.issuer construct in the directoryName field. The subject.issuerSerial.issuerUID field MUST only be used if the subject's PKC contains an issuerUniqueID field.

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

If the subject field uses the subjectName option then only one name should be present.

For all GeneralName fields in this profile the otherName, x400Address, ediPartyName and registeredId options MUST NOT be used

Any protocol which uses 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 contain one and only one GeneralName which must contain its value in the directoryName field. This means that all AC issuers MUST have non-NULL X.500 names.

4.3.4 validity

The validity field specifies the period for which the AC issuer expects that the binding between the subject and the attributes fields will be valid. There is no implied guarantee that the binding will actually be valid at any given moment during the validity period.

GeneralizedTime encoding is restricted as specified in [PKIX-1] 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.

4.3.5 signature

Contains the algorithm identifier used to validate the AC signature.

This MUST be one of:

algorithm	OID
rsaWithSHA1	tbs - from PKIX/CMS
rsaWithMD5	tbs
dsaWithSHA1	tbs
dsaWithMD5	tbs

dsaWithSHA1 MUST be supported by all AC users. The other algorithms SHOULD be supported.

4.3.6 serial

For any given AC issuer, the issuer/serial pair MUST form a unique combination, even if ACs are very short-lived (one second is the shortest possible validity according to the above).

AC issuers MUST force the serial number 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 done by adding a leading (leftmost) `00'H octet if necessary. This removes a potential ambiguity in mapping between an string of octets and a serial number.

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, e.g. they needn't be monotonically increasing with time.

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 SET OF Attribute. For a given AC each attribute type in the set 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.

Standard attribute types are defined in section 4.5.

4.3.8 issuerUID

This field MUST NOT be used.

INTERNET-DRAFT AC PROF August 20 1998

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 which MAY be used with this profile. An AC which 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 which contains additional non-critical extensions still conforms.

4.4 Extensions.

4.4.1 Restrictions

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 version can only be supported via the use of a critical extension.

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

In addition an attribute type which occurs in the attributes field MUST NOT occur in the restrictions field (if present). This ensures that the entire AC contains only one instance of any attribute type at the expense of forcing the definition of new OIDs for some 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 identify individuals. This makes the use of the subject 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 impossible to derive the AC owner's identity from the audit identity value.

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 can presumably identify the AC owner in cases where mis-behaviour is detected.

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.

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 which 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".

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

4.4.3 AC Targeting

In order to allow that an AC is "targeted" the delegation information extension specifies a number of servers/services. The intent is that the AC should only be usable at these servers/services - an (honest) AC verifier who is not amongst the named servers/services MUST reject the AC.

This extension also controls delegation.

INTERNET-DRAFT AC PROF August 20 1998

If this extension is not present then the AC is not delegatable. Any server which receives the AC such that the subject 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 delegate information (delegates field). If the "direct check" or any of the "delegate" checks (see below) pass then the "targeting check" as a whole is successful.

Though the rules given below look complex, they aren't - the effect is that the AC owner can send to any valid target which can then only delegate to targets which are in one of the same delegate sets as itself.

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

```
DelegationInfo ::= SEQUENCE {
     owner     CHOICE {
          baseCertificateID   [0]  IssuerSerial,
          subjectName         [1]  GeneralNames
     },
     targets   Targets             OPTIONAL,
     delegates SEQUENCE OF Targets OPTIONAL
}
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 OID choice and a value of {ietf-ac-extensions 4 1}.

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.

A delegate 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 delegate 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 delegate 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 delegated 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 delegate 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.

```
OID            { ietf-ac-extensions 4 }
syntax         DelegationInfo
criticality    must be TRUE
```

4.4.4 authorityKeyIdentifier

The authorityKeyIdentifier extension as profiled in [PKIX-1] MAY be used to assist the AC verifier in checking the signature of the AC. The [PKIX-1] description should be read as if "CA" meant "AC issuer". As with PKCs this extension SHOULD be included in ACs.

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

4.5 Attribute Types

<<it'd be much nicer to inherit all of the attribute definitions instead of making new syntax - any suitable candidates?>>

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 which 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.

```
        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).

```
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.

```
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.

```
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 be more useful to defined OS specific group attribute types which map to UNIX gids or NT SIDs?>>

```
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.

```
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.

```
OID        { id-aa-securityLabel }
          { iso(1) member-body(2) us(840) rsadsi(113549)
          pkcs(1) pkcs-9(9) smime(16) id-aa(2) 2}
syntax     ESSSecurityLabel - imported from [ESS]
values     Multiple allowed
```

4.5.7 EncryptedAttributes

Where an AC will be carried in clear within an application protocol or where an AC which may be delegated 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 encrytpedAttributes 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 which 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 serial 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 serial 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 which gives the same result is fine):

1. Identify the sets of attributes which 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

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

5. AttributeCertificate Validation

This section describes a basic set of rules which 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 time of evaluation MUST be within validity (if the evaluation time is equal to either notBefore or notAfter then the AC is timely, i.e. this check succeeds)
2. the signature must be valid - based on a PKC for the AC issuer
3. the AC validity.notBefore must be within the validity period of the AC issuer's PKC
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. the AC issuer MUST be explicitly trusted - this is NOT the same as the AC having a valid PKC - the AC verifier will require some additional configuration/parameterisation in order to determine this
7. if the AC contains any "unsupported" critical extensions then the AC MUST be rejected.

"Support" for an extension in this context means:

1. the AC verifier MUST be able to parse the extension value
2. where the extension value SHOULD cause the AC to be rejected, the AC verifier MUST reject the AC

Additional Checks:

1. The AC MAY be rejected on the basis of further AC verifier configuration, for example an AC verifier may be configured to reject ACs which contain or lack certain attribute types
2. If the AC verifier provides an interface which 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. Conformance

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

A fully conformant AC issuer MUST be able to produce all of the attribute types and extensions specified above. A fully conformant AC verifier MUST "support" all of the attribute types and extensions specified above. "Support" in the previous sentence means more than just parsing - it means that the AC verifier (which is part of a target) 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 fully conformant AC issuer is responsible to ensure that no AC produced could be accepted by a simply conformant AV verifier in such a way as to cause a security breach.

<<dunno if that can happen but I should think about it>>

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

1. always use the baseCertificateID subject name alternative
2. are never post-dated
3. can contain AccessIdentity, Group and/or Role attributes with multiple values
4. do not contain any other attributes which cannot safely be ignored by an AC verifier
5. can contain the AuthorityKeyIdentifier extension
6. contain no critical extensions (and hence is not delegatable)
7. do not contain encrypted attributes

Simple conformance for an AC verifier means support for the validation of ACs which are produced by simply conformant AC issuers. A simply comformant AC verifier can ignore the presence of any unsupported attributes or

extensions (of course it must reject all ACs which contain critical extensions) and need only make the values of the above attributes available to applications.

7. Security Considerations

tbs

8. References

[CMS] Housley, R., "Cryptographic Message Syntax", draft-ietf-smime-cms-05.txt, May 1998.

[ESS] Hoffman, P., "Enhanced Security Services for S/MIME", draft-ietf-smime-ess-05.txt, April 1998.

[PKIX-1] Housley, R., Ford, W., Polk, T, & Solo, D., "Internet Public Key Infrastructure - X.509 Certificate and CRL profile", draft-ietf-pkix-ipki-part1-07.txt, March 1998.

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

Author's Address

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

tel: +353-1-676-9089
email: stephen.farrell@sse.ie

Appendix 1: Samples

tbs

Appendix 2: "Compilable" ASN.1 Module

tbs