Network Working Group
Request for Comments: 5035
Updates: 2634
Category: Standards Track

J. Schaad
Soaring Hawk Consulting

August 2007

Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility

Status of This Memo

Abstract

In the original Enhanced Security Services for S/MIME document (RFC 2634), a structure for cryptographically linking the certificate to be used in validation with the signature was introduced; this structure was hardwired to use SHA-1. This document allows for the structure to have algorithm agility and defines a new attribute for this purpose.

Table of Contents

1. Introduction

In the original Enhanced Security Services (ESS) for S/MIME document [ESS], a structure for cryptographically linking the certificate to be used in validation with the signature was defined. This structure, called ESSCertID, identifies a certificate by its hash. The structure is hardwired to use a SHA-1 hash value. The recent attacks on SHA-1 require that we define a new attribute that allows for the use of different algorithms. This document performs that task.

This document defines the structure ESSCertIDv2 along with a new attribute SigningCertificateV2, which uses the updated structure. This document allows for the structure to have algorithm agility by including an algorithm identifier and defines a new signed attribute to use the new structure.

This document specifies the continued use of ESSCertID to ensure compatibility when SHA-1 is used for certificate disambiguation.

1.1. Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Updates to RFC 2634

This document updates Section 5.4 of RFC 2634. Once the updates are applied, the revised section will have the following structure:

5.4 Signing Certificate Attribute Definitions

5.4.1 Signing Certificate Attribute Definition Version 2

5.4.1.1 Certificate Identification Version 2

5.4.2 Signing Certificate Attribute Definition Version 1

5.4.2.1 Certificate Identification Version 1

In addition, the ASN.1 module in Appendix A is replaced.

2. Replace Section 5.4 'Signing Certificate Attribute Definitions'

5.4 Signing Certificate Attribute Definitions

The signing certificate attribute is designed to prevent simple substitution and re-issue attacks, and to allow for a restricted set of certificates to be used in verifying a signature.

Two different attributes exist for this due to a flaw in the original design. The only substantial difference between the two attributes is that SigningCertificateV2 allows for hash algorithm agility, while SigningCertificate forces the use of the SHA-1 hash algorithm. With the recent advances in the ability to create hash collisions for SHA-1, it is wise to move forward sooner rather than later.

When the SHA-1 hash function is used, the SigningCertificate attribute MUST be used. The SigningCertificateV2 attribute MUST be used if any algorithm other than SHA-1 is used and SHOULD NOT be used for SHA-1. Applications SHOULD recognize both attributes as long as they consider SHA-1 able to distinguish between two different certificates, (i.e., the possibility of a collision is sufficiently low). If both attributes exist in a single message, they are independently evaluated.

Four cases exist that need to be taken into account when using this attribute for correct processing:

1. Signature validates and the hashes match: This is the success case.

2. Signature validates and the hashes do not match: In this case, the certificate contained the correct public key, but the certificate containing the public key is not the one that the signer intended to be used. In this case the application should attempt a search for a different certificate with the same public key and for which the hashes match. If no such certificate can be found, this is a failure case.

3. Signature fails validation and the hashes match: In this case, it can be assumed that the signature has been modified in some fashion. This is a failure case.

4. Signature fails validation and the hashes do not match: In this case, it can be either that the signature has been modified, or that the wrong certificate has been used. Applications should attempt a search for a different certificate that matches the hash value in the attribute and use the new certificate to retry the signature validation.

3. Insert New Section 5.4.1 'Signing Certificate Attribute Definition Version 2'

5.4.1 Signing Certificate Attribute Definition Version 2

The signing certificate attribute is designed to prevent the simple substitution and re-issue attacks, and to allow for a restricted set of certificates to be used in verifying a signature.

SigningCertificateV2 is identified by the OID:

```
id-aa-signingCertificateV2 OBJECT IDENTIFIER ::= { iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) id-aa(2) 47 }
```

The attribute has the ASN.1 definition:

```
SigningCertificateV2 ::=  SEQUENCE {
    certs        SEQUENCE OF ESSCertIDv2,
    policies     SEQUENCE OF PolicyInformation OPTIONAL
}
```

certs
: contains the list of certificates that are to be used in validating the message. The first certificate identified in the sequence of certificate identifiers MUST be the certificate used to verify the signature. The encoding of the ESSCertIDv2 for this certificate SHOULD include the issuerSerial field. If other constraints ensure that issuerAndSerialNumber will be present in the SignerInfo, the issuerSerial field MAY be omitted. The certificate identified is used during the signature verification process. If the hash of the certificate does not match the certificate used to verify the signature, the signature MUST be considered invalid.

  If more than one certificate is present, subsequent certificates limit the set of certificates that are used during validation. Certificates can be either attribute certificates (limiting authorizations) or public key certificates (limiting path validation). The issuerSerial field (in the ESSCertIDv2 structure) SHOULD be present for these certificates, unless the client who is validating the signature is expected to have easy access to all the certificates required for validation. If only the signing certificate is present in the sequence, there are no restrictions on the set of certificates used in validating the signature.

policies
contains a sequence of policy information terms that identify those certificate policies that the signer asserts apply to the certificate, and under which the certificate should be relied upon. This value suggests a policy value to be used in the relying party's certification path validation. The definition of PolicyInformation can be found in [RFC3280].

If present, the SigningCertificateV2 attribute MUST be a signed attribute; it MUST NOT be an unsigned attribute. CMS defines SignedAttributes as a SET OF Attribute. A SignerInfo MUST NOT include multiple instances of the SigningCertificateV2 attribute. CMS defines the ASN.1 syntax for the signed attributes to include attrValues SET OF AttributeValue. A SigningCertificateV2 attribute MUST include only a single instance of AttributeValue. There MUST NOT be zero or multiple instances of AttributeValue present in the attrValues SET OF AttributeValue.

4. Insert New Section 5.4.1.1 'Certificate Identification Version 2'

Insert the following text as a new section.

5.4.1.1 Certificate Identification Version 2

The best way to identify certificates is an often-discussed issue. The ESSCertIDv2 structure supplies two different fields that are used for this purpose.

The hash of the entire certificate allows for a verifier to check that the certificate used in the verification process was the same certificate the signer intended. Hashes are convenient in that they are frequently used by certificate stores as a method of indexing and retrieving certificates as well. The use of the hash is required by this structure since the detection of substituted certificates is based on the fact they would map to different hash values.

The issuer/serial number pair is the method of identification of certificates used in [RFC3280]. That document imposes a restriction for certificates that the issuer distinguished name must be present. The issuer/serial number pair would therefore normally be sufficient to identify the correct signing certificate. (This assumes the same issuer name is not reused from the set of trust anchors.) The issuer/serial number pair can be stored in the sid field of the SignerInfo object. However, the sid field is not covered by the signature. In the cases where the issuer/serial number pair is not used in the sid or the issuer/serial number pair needs to be signed, it SHOULD be placed in the issuerSerial field of the ESSCertIDv2 structure.

Attribute certificates and additional public key certificates containing information do not have an issuer/serial number pair represented anywhere in a SignerInfo object. When an attribute certificate or an additional public key certificate is not included in the SignedData object, it becomes much more difficult to get the correct set of certificates based only on a hash of the certificate. For this reason, these certificates SHOULD be identified by the IssuerSerial object.

This document defines a certificate identifier as:

```
ESSCertIDv2 ::=  SEQUENCE {
    hashAlgorithm           AlgorithmIdentifier
           DEFAULT {algorithm id-sha256},
    certHash                Hash,
    issuerSerial            IssuerSerial OPTIONAL
}

Hash ::= OCTET STRING

IssuerSerial ::= SEQUENCE {
    issuer                  GeneralNames,
    serialNumber            CertificateSerialNumber
}
```

The fields of ESSCertIDv2 are defined as follows:

hashAlgorithm
: contains the identifier of the algorithm used in computing certHash.

certHash
: is computed over the entire DER-encoded certificate (including the signature) using the SHA-1 algorithm.

issuerSerial
: holds the identification of the certificate. The issuerSerial would normally be present unless the value can be inferred from other information (e.g., the sid field of the SignerInfo object).

The fields of IssuerSerial are defined as follows:

issuer
: contains the issuer name of the certificate. For non-attribute certificates, the issuer MUST contain only the issuer name from the certificate encoded in the directoryName choice of GeneralNames. For attribute certificates, the issuer MUST contain the issuer name field from the attribute certificate.

serialNumber
holds the serial number that uniquely identifies the certificate for the issuer.

5. Insert New Section 5.4.2 'Signing Certificate Attribute Definition Version 1'

(Note: This section does not present new material. This section contains the original contents of Section 5.4 in [ESS], which are retained with minor changes in this specification to achieve backwards compatibility.)

Insert the following text as a new section.

5.4.2 Signing Certificate Attribute Definition Version 1

The signing certificate attribute is designed to prevent the simple substitution and re-issue attacks, and to allow for a restricted set of certificates to be used in verifying a signature.

The definition of SigningCertificate is

```
SigningCertificate ::=  SEQUENCE {
    certs        SEQUENCE OF ESSCertID,
    policies     SEQUENCE OF PolicyInformation OPTIONAL
}

id-aa-signingCertificate OBJECT IDENTIFIER ::= { iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) id-aa(2) 12 }
```

The first certificate identified in the sequence of certificate identifiers MUST be the certificate used to verify the signature. The encoding of the ESSCertID for this certificate SHOULD include the issuerSerial field. If other constraints ensure that issuerAndSerialNumber will be present in the SignerInfo, the issuerSerial field MAY be omitted. The certificate identified is used during the signature verification process. If the hash of the certificate does not match the certificate used to verify the signature, the signature MUST be considered invalid.

If more than one certificate is present in the sequence of ESSCertIDs, the certificates after the first one limit the set of certificates that are used during validation. Certificates can be either attribute certificates (limiting authorizations) or public key certificates (limiting path validation). The issuerSerial field (in the ESSCertID structure) SHOULD be present for these certificates, unless the client who is validating the signature is expected to have

easy access to all the certificates required for validation. If only the signing certificate is present in the sequence, there are no restrictions on the set of certificates used in validating the signature.

The sequence of policy information terms identifies those certificate policies that the signer asserts apply to the certificate, and under which the certificate should be relied upon. This value suggests a policy value to be used in the relying party's certification path validation.

If present, the SigningCertificate attribute MUST be a signed attribute; it MUST NOT be an unsigned attribute. Cryptographic Message Syntax (CMS) defines SignedAttributes as a SET OF Attribute. A SignerInfo MUST NOT include multiple instances of the SigningCertificate attribute. CMS defines the ASN.1 syntax for the signed attributes to include attrValues SET OF AttributeValue. A SigningCertificate attribute MUST include only a single instance of AttributeValue. There MUST NOT be zero or multiple instances of AttributeValue present in the attrValues SET OF AttributeValue.

6. Insert New Section 5.4.2.1 'Certificate Identification Version 1'

(Note: This section does not present new material. This section contains the original contents of Section 5.4 in [ESS], which are retained with minor changes in this specification to achieve backwards compatibility.)

Delete old Section 5.4.1

Insert the following as new text

5.4.2.1 Certificate Identification Version 1

Certificates are uniquely identified using the information in the ESSCertID structure. Discussion can be found in Section 5.4.1.1.

This document defines a certificate identifier as:

```
ESSCertID ::=  SEQUENCE {
     certHash                 Hash,
     issuerSerial             IssuerSerial OPTIONAL
}
```

The fields of ESSCertID are defined as follows:

certHash
: is computed over the entire DER-encoded certificate (including the signature).

issuerSerial
: holds the identification of the certificate. This field would normally be present unless the value can be inferred from other information (e.g., the sid field of the SignerInfo object).

The fields of IssuerSerial are discussed in Section 5.4.1.1

7. Security Considerations

This document is designed to address the security issue of a substituted certificate used by the validator. If a different certificate is used by the validator than the signer, the validator may not get the correct result. An example of this would be that the original certificate was revoked and a new certificate with the same public key was issued for a different individual. Since the issuer/serial number field is not protected, the attacker could replace this and point to the new certificate and validation would be successful.

The attributes defined in this document are to be placed in locations that are protected by the signature. This attribute does not provide any additional security if placed in an unsigned or un-authenticated location.

The attributes defined in this document permit a signer to select a hash algorithm to identify a certificate. A poorly selected hash algorithm may provide inadequate protection against certificate substitution or result in denial of service for this protection. By employing the attributes defined in this specification with the same hash algorithm used for message signing, the sender can ensure that these attributes provide commensurate security.

Since recipients must support the hash algorithm to verify the signature, selecting the same hash algorithm also increases the likelihood that the hash algorithm is supported in the context of certificate identification. Note that an unsupported hash algorithm for certificate identification does not preclude validating the message but does deny the message recipient protection against certificate substitution.

To ensure that legacy implementations are provided protection against certificate substitution, clients are permitted to include both ESScertID and ESScertIDv2 in the same message. Since these

attributes are generated and evaluated independently, the contents could conceivably be in conflict. Specifically, where a signer has multiple certificates containing the same public key, the two attributes could specify different signing certificates. The result of signature processing may vary depending on which certificate is used to validate the signature.

Recipients that attempt to evaluate both attributes may choose to reject such a message.

8. Normative References

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

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

[RFC3280] Housley, R., Ford, W., Polk, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[RFC3852] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

[UTF8] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

Appendix A. ASN.1 Module

Replace the ASN.1 module in RFC 2634 with this one.

```
ExtendedSecurityServices-2006
      { iso(1) member-body(2) us(840) rsadsi(113549)
        pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-ess-2006(30) }
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
IMPORTS
-- Cryptographic Message Syntax (CMS) [RFC3852]
    ContentType, IssuerAndSerialNumber, SubjectKeyIdentifier
       FROM CryptographicMessageSyntax2004 { iso(1) member-body(2)
                us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                modules(0) cms-2004(24)}
-- PKIX Certificate and CRL Profile, Section A.1 Explicity Tagged Module
--  1988 Syntax             [RFC3280]
    AlgorithmIdentifier, CertificateSerialNumber
       FROM PKIX1Explicit88 { iso(1) identified-organization(3) dod(6)
                internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
                id-pkix1-explicit(18) }

-- PKIX Certificate and CRL Profile, Sec A.2 Implicitly Tagged Module,
--  1988 Syntax             [RFC3280]
    PolicyInformation, GeneralNames
          FROM PKIX1Implicit88 {iso(1) identified-organization(3) dod(6)
                internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
                id-pkix1-implicit(19)};

-- Extended Security Services
-- The construct "SEQUENCE SIZE (1..MAX) OF" appears in several ASN.1
-- constructs in this module.  A valid ASN.1 SEQUENCE can have zero or
-- more entries.  The SIZE (1..MAX) construct constrains the SEQUENCE to
-- have at least one entry.  MAX indicates the upper bound is
-- unspecified.  Implementations are free to choose an upper bound that
-- suits their environment.

-- UTF8String ::= [UNIVERSAL 12] IMPLICIT OCTET STRING

-- The contents are formatted as described in [UTF8]

-- Section 2.7

ReceiptRequest ::= SEQUENCE {
  signedContentIdentifier ContentIdentifier,
  receiptsFrom ReceiptsFrom,
  receiptsTo SEQUENCE SIZE (1..ub-receiptsTo) OF GeneralNames
}
```

```
ub-receiptsTo INTEGER ::= 16

id-aa-receiptRequest OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 1}

ContentIdentifier ::= OCTET STRING

id-aa-contentIdentifier OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 7}

ReceiptsFrom ::= CHOICE {
 allOrFirstTier [0] AllOrFirstTier, -- formerly "allOrNone [0]AllOrNone"
 receiptList [1] SEQUENCE OF GeneralNames
}

AllOrFirstTier ::= INTEGER { -- Formerly AllOrNone
 allReceipts (0),
 firstTierRecipients (1)
}

-- Section 2.8

Receipt ::= SEQUENCE {
  version ESSVersion,
  contentType ContentType,
  signedContentIdentifier ContentIdentifier,
  originatorSignatureValue OCTET STRING
}

id-ct-receipt OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
   rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-ct(1) 1}

ESSVersion ::= INTEGER  { v1(1) }

-- Section 2.9

ContentHints ::= SEQUENCE {
  contentDescription UTF8String (SIZE (1..MAX)) OPTIONAL,
  contentType ContentType
}

id-aa-contentHint OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
    rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 4}

-- Section 2.10

MsgSigDigest ::= OCTET STRING
```

```
id-aa-msgSigDigest OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 5}

-- Section 2.11

ContentReference ::= SEQUENCE {
  contentType ContentType,
  signedContentIdentifier ContentIdentifier,
  originatorSignatureValue OCTET STRING
}

id-aa-contentReference   OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 10 }

-- Section 3.2

ESSSecurityLabel ::= SET {
  security-policy-identifier SecurityPolicyIdentifier,
  security-classification SecurityClassification OPTIONAL,
  privacy-mark ESSPrivacyMark OPTIONAL,
  security-categories SecurityCategories OPTIONAL
}

id-aa-securityLabel OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 2}
SecurityPolicyIdentifier ::= OBJECT IDENTIFIER

SecurityClassification ::= INTEGER {
  unmarked (0),
  unclassified (1),
  restricted (2),
  confidential (3),
  secret (4),
  top-secret (5)
}(0..ub-integer-options)

ub-integer-options INTEGER ::= 256

ESSPrivacyMark ::= CHOICE {
    pString      PrintableString (SIZE (1..ub-privacy-mark-length)),
    utf8String   UTF8String (SIZE (1..MAX))
}

ub-privacy-mark-length INTEGER ::= 128

SecurityCategories ::= SET SIZE (1..ub-security-categories) OF
        SecurityCategory
```

```
ub-security-categories INTEGER ::= 64

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

--Note: The aforementioned SecurityCategory syntax produces identical
--hex encodings as the following SecurityCategory syntax that is
--documented in the X.411 specification:
--
--SecurityCategory ::= SEQUENCE {

--     type  [0]  SECURITY-CATEGORY,
--     value [1]  ANY DEFINED BY type }
--
--SECURITY-CATEGORY MACRO ::=
--BEGIN
--TYPE NOTATION ::= type | empty
--VALUE NOTATION ::= value (VALUE OBJECT IDENTIFIER)
--END

-- Section 3.4

EquivalentLabels ::= SEQUENCE OF ESSSecurityLabel

id-aa-equivalentLabels OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 9}

-- Section 4.4

MLExpansionHistory ::= SEQUENCE
      SIZE (1..ub-ml-expansion-history) OF MLData

id-aa-mlExpandHistory OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 3 }

ub-ml-expansion-history INTEGER ::= 64  MLData ::= SEQUENCE {
  mailListIdentifier EntityIdentifier,
  expansionTime GeneralizedTime,
  mlReceiptPolicy MLReceiptPolicy OPTIONAL
}

EntityIdentifier ::= CHOICE {
  issuerAndSerialNumber IssuerAndSerialNumber,
  subjectKeyIdentifier SubjectKeyIdentifier
}
```

```
MLReceiptPolicy ::= CHOICE {
  none [0] NULL,
  insteadOf [1] SEQUENCE SIZE (1..MAX) OF GeneralNames,
  inAdditionTo [2] SEQUENCE SIZE (1..MAX) OF GeneralNames
}

-- Section 5.4

SigningCertificate ::=  SEQUENCE {
    certs        SEQUENCE OF ESSCertID,
    policies     SEQUENCE OF PolicyInformation OPTIONAL
}

id-aa-signingCertificate OBJECT IDENTIFIER ::= { iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) id-aa(2) 12 }

SigningCertificateV2 ::=  SEQUENCE {
    certs        SEQUENCE OF ESSCertIDv2,
    policies     SEQUENCE OF PolicyInformation OPTIONAL
}

id-aa-signingCertificateV2 OBJECT IDENTIFIER ::= { iso(1)
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) id-aa(2) 47 }

id-sha256  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101)
    csor(3) nistalgorithm(4) hashalgs(2) 1 }

ESSCertIDv2 ::= SEQUENCE {
     hashAlgorithm           AlgorithmIdentifier
            DEFAULT {algorithm id-sha256},
     certHash                Hash,
     issuerSerial            IssuerSerial OPTIONAL
}

ESSCertID ::=  SEQUENCE {
     certHash                 Hash,
     issuerSerial             IssuerSerial OPTIONAL
}

Hash ::= OCTET STRING  IssuerSerial ::= SEQUENCE {
     issuer                   GeneralNames,
     serialNumber             CertificateSerialNumber
}

END
```

-- of ExtendedSecurityServices-2006

Author’s Address

Jim Schaad
Soaring Hawk Consulting
PO Box 675
Gold Bar, WA 98251

EMail: jimsch@exmsft.com

Full Copyright Statement

Intellectual Property