Network Working Group
Internet-Draft
Updates: 3370, 3851, 3852, 4108, 5035, 5083, 5084 (if approved)
Expires: May 11, 2008

P. Hoffman
VPN Consortium
J. Schaad
Soaring Hawk Consulting

November 8, 2007

New ASN.1 Modules for CMS and S/MIME
draft-hoffman-cms-new-asn1-00.txt

Status of this Memo

Copyright Notice

Abstract

The Cryptographic Message Syntax (CMS) format, and many associated formats, are expressed using ASN.1. The current ASN.1 modules conform to the 1988 version of ASN.1. This document updates those ASN.1 modules to conform to the 2002 version of ASN.1. There are no bits-on-the-wire changes to any of the formats; this is simply a change to the syntax.

Table of Contents

1. Introduction 3
1.1. Issues 3
1.1.1. More Modules To Be Added 3
1.1.2. Algorithm Structure 4
1.1.3. Module OIDs Changing 4
2. ASN.1 Module for RFC 3370 4
3. ASN.1 Module for RFC 3851 7
4. ASN.1 Module for RFC 3852 9
5. ASN.1 Module for RFC 4108 18
6. ASN.1 Module for RFC 5035 22
7. ASN.1 Module for RFC 5083 28
8. ASN.1 Module for RFC 5084 29
9. Security Considerations 29
10. Normative References 30
Authors' Addresses 30
Intellectual Property and Copyright Statements 32

1. Introduction

Some developers would like the IETF to use the latest version of ASN.1 in its standards. Most of the RFCs that relate to security protocols still use ASN.1 from the 1988 standard, which has been deprecated. This is particularly true for the standards that relate to PKIX, CMS, and S/MIME.

This document updates the following RFCs to use ASN.1 modules that conform to the 2002 version of ASN.1 [ASN1-2002]. Note that not all the modules are updated; some are included to simply make the set compete.

- RFC 3370, CMS Algorithms [RFC3370]
- RFC 3851, S/MIME Version 3.1 Message Specification [RFC3851]
- RFC 3852, CMS main [RFC3852]
- RFC 4108, Using CMS to Protect Firmware Packages [RFC4108]
- RFC 5035, Enhanced Security Services (ESS) [RFC5035]
- RFC-to-be 5083, CMS Authenticated-Enveloped-Data Content Type [RFC5083]
- RFC-to-be 5084, Using AES-CCM and AES-GCM Authenticated Encryption in CMS [RFC5084]

Note that some of the modules in this document get some of their definitions from places different than the modules in the original RFCs. The idea is that these modules, when combined with the modules in [NEW-PKIX] can stand on their own and do not need to import definitions from anywhere else.

1.1. Issues

This section will be removed before final publication.

1.1.1. More Modules To Be Added

There are many modules from standards-track RFCs that are not listed in this document or the companion document on PKIX. We will discuss with the two communities which modules are appropriate for the two documents. We will also consider making "super-modules", individual modules which might update multiple RFCs at one time. We may also add objects to some of the modules.

1.1.2. Algorithm Structure

Algorithms are currently not defined here. We need to discuss what structure we want for algorithm objects. Currently, we just do "parameter, OID", but we could add more. Because we don't know what the final structure is, the object sets in the various modules are commented out. We will fix this before finishing this project.

1.1.3. Module OIDs Changing

The OIDs given in the modules in this version of the document are the same as the OIDs from the original modules, even though some of the modules have changed syntax. That is clearly incorrect. In a later version of this document, we will change the OIDs for every changed module.

2. ASN.1 Module for RFC 3370

```
CryptographicMessageSyntaxAlgorithms
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) modules(0) cmsalg-2001(16) }
DEFINITIONS IMPLICIT TAGS ::=
BEGIN


-- The following is easier than importing it from PKIX1Explicit88,
--     and makes the module stand-alone
ALGORITHM ::= TYPE-IDENTIFIER

-- Algorithm Identifiers

sha-1 OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
   oiw(14) secsig(3) algorithm(2) 26 }

md5 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
   rsadsi(113549) digestAlgorithm(2) 5 }

id-dsa OBJECT IDENTIFIER ::=  { iso(1) member-body(2) us(840)
   x9-57(10040) x9cm(4) 1 }

id-dsa-with-sha1 OBJECT IDENTIFIER ::=  { iso(1) member-body(2)
   us(840) x9-57(10040) x9cm(4) 3 }

rsaEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 1 }

md5WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1)
```

```
     member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 4 }

  sha1WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1)
     member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }

  dh-public-number OBJECT IDENTIFIER ::= { iso(1) member-body(2)
     us(840) ansi-x942(10046) number-type(2) 1 }

  id-alg-ESDH OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
     rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 5 }

  id-alg-SSDH OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
     rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 10 }

  id-alg-CMS3DESwrap OBJECT IDENTIFIER ::= { iso(1) member-body(2)
     us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 6 }

  id-alg-CMSRC2wrap OBJECT IDENTIFIER ::= { iso(1) member-body(2)
     us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 7 }

  des-ede3-cbc OBJECT IDENTIFIER ::= { iso(1) member-body(2)
     us(840) rsadsi(113549) encryptionAlgorithm(3) 7 }

  rc2-cbc OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
     rsadsi(113549) encryptionAlgorithm(3) 2 }

  hMAC-SHA1 OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
     dod(6) internet(1) security(5) mechanisms(5) 8 1 2 }

  id-PBKDF2 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
     rsadsi(113549) pkcs(1) pkcs-5(5) 12 }

  -- Public Key Types

  Dss-Pub-Key ::= INTEGER  -- Y

  RSAPublicKey ::= SEQUENCE {
      modulus INTEGER,  -- n
      publicExponent INTEGER }  -- e

  DHPublicKey ::= INTEGER  -- y = g^x mod p


  -- Signature Value Types

  Dss-Sig-Value ::= SEQUENCE {
      r INTEGER,
      s INTEGER }
```

```
  -- Algorithm Identifier Parameter Types

  Dss-Parms ::= SEQUENCE {
      p INTEGER,
      q INTEGER,
      g INTEGER }

  DHDomainParameters ::= SEQUENCE {
      p INTEGER,  -- odd prime, p=jq +1
      g INTEGER,  -- generator, g
      q INTEGER,  -- factor of p-1
      j INTEGER OPTIONAL,  -- subgroup factor
      validationParms ValidationParms OPTIONAL }

  ValidationParms ::= SEQUENCE {
      seed BIT STRING,
      pgenCounter INTEGER }

  KeyWrapAlgorithm ::=
      AlgorithmIdentifier {{SupportedKeyWrapAlgorithms}}

  SupportedKeyWrapAlgorithms ALGORITHM ::= { ... }

  RC2wrapParameter ::= RC2ParameterVersion

  RC2ParameterVersion ::= INTEGER

  CBCParameter ::= IV

  IV ::= OCTET STRING  -- exactly 8 octets

  RC2CBCParameter ::= SEQUENCE {
      rc2ParameterVersion INTEGER (1..256),
      iv OCTET STRING  }  -- exactly 8 octets

  algid-hMAC-SHA1 ALGORITHM ::= { NULL IDENTIFIED BY hMAC-SHA1 }

  -- Another way to do the following would be:
  -- alg-hMAC-SHA1 AlgorithmIdentifier{{PBKDF2-PRFs}} ::=
      { algorithm hMAC-SHA1, parameters NULL:NULL }

  PBKDF2-PRFsAlgorithmIdentifier ::= AlgorithmIdentifier{{PBKDF2-PRFs}}

  alg-hMAC-SHA1 PBKDF2-PRFsAlgorithmIdentifier ::=
      { algorithm hMAC-SHA1, parameters NULL:NULL }

  PBKDF2-SaltSources ALGORITHM ::= { ... }
```

```
  PBKDF2-PRFs ALGORITHM ::= { algid-hMAC-SHA1, ... }

  PBKDF2-SaltSourcesAlgorithmIdentifier ::=
      AlgorithmIdentifier {{PBKDF2-SaltSources}}

  PBKDF2-params ::= SEQUENCE {
      salt CHOICE {
          specified OCTET STRING,
          otherSource PBKDF2-SaltSourcesAlgorithmIdentifier },
      iterationCount INTEGER (1..MAX),
      keyLength INTEGER (1..MAX) OPTIONAL,
      prf PBKDF2-PRFsAlgorithmIdentifier DEFAULT
              alg-hMAC-SHA1 }

  AlgorithmIdentifier { ALGORITHM:InfoObjectSet } ::= SEQUENCE {
      algorithm ALGORITHM.&id({InfoObjectSet}),
      parameters ALGORITHM.&Type({InfoObjectSet}{@algorithm}) OPTIONAL }

  END
```

3. ASN.1 Module for RFC 3851

```
  SecureMimeMessageV3dot1
        { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
        smime(16) modules(0) msg-v3dot1(21) }
  DEFINITIONS IMPLICIT TAGS ::=
  BEGIN

  IMPORTS

  SubjectKeyIdentifier, IssuerAndSerialNumber, RecipientKeyIdentifier
  FROM CryptographicMessageSyntax
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) cms-2004(24) };



  --  id-aa is the arc with all new authenticated and unauthenticated
  --  attributes produced the by S/MIME Working Group

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

  -- S/MIME Capabilities provides a method of broadcasting the symmetric
  -- capabilities understood.  Algorithms SHOULD be ordered by
  -- preference and grouped by type
```

```
  smimeCapabilities OBJECT IDENTIFIER ::=
     {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) 15}

  SMIME-CAPS ::= TYPE-IDENTIFIER

  SMIMECapability ::= SEQUENCE {
     capabilityID         SMIME-CAPS.
                &id({SMimeCapsSet}),
     parameters           SMIME-CAPS.
                &Type({SMimeCapsSet}{@capabilityID}) OPTIONAL }

  SMimeCapsSet SMIME-CAPS ::= { ... }

  SMIMECapabilities ::= SEQUENCE OF SMIMECapability

  -- Encryption Key Preference provides a method of broadcasting the
  -- preferred encryption certificate.

  id-aa-encrypKeyPref OBJECT IDENTIFIER ::= {id-aa 11}

  SMIMEEncryptionKeyPreference ::= CHOICE {
     issuerAndSerialNumber   [0] IssuerAndSerialNumber,
     receipentKeyId          [1] RecipientKeyIdentifier,
     subjectAltKeyIdentifier [2] SubjectKeyIdentifier
  }


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

  id-cap  OBJECT IDENTIFIER ::= { id-smime 11 }

  -- The preferBinaryInside indicates an ability to receive messages
  -- with binary encoding inside the CMS wrapper

  id-cap-preferBinaryInside  OBJECT IDENTIFIER ::= { id-cap 1 }

  --  The following list the OIDs to be used with S/MIME V3

  -- Signature Algorithms Not Found in [CMSALG]
  --
  -- md2WithRSAEncryption OBJECT IDENTIFIER ::=
  --    {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1)
  --     2}
  --
  -- Other Signed Attributes
  --
  -- signingTime OBJECT IDENTIFIER ::=
```

```
  --    {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
  --     5}
  --    See [CMS] for a description of how to encode the attribute
  --    value.

  SMIMECapabilitiesParametersForRC2CBC ::= INTEGER
  --        (RC2 Key Length (number of bits))

  END
```

4. ASN.1 Module for RFC 3852

```
   CryptographicMessageSyntax2004
       { iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs-9(9) smime(16) modules(0) cms-2004(24) }
   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN

   IMPORTS

   ALGORITHM, Certificate, CertificateList, CertificateSerialNumber,
       Name
   FROM PKIX1Explicit88
       { iso(1) identified-organization(3) dod(6) internet(1)
       security(5) mechanisms(5) pkix(7) id-mod(0)
       id-pkix1-explicit(18) }

   AttributeCertificate
   FROM PKIXAttributeCertificate
       { iso(1) identified-organization(3) dod(6) internet(1)
       security(5) mechanisms(5) pkix(7) id-mod(0)
       id-mod-attribute-cert(12) }

   AttributeCertificateV1
   FROM AttributeCertificateVersion1
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
       smime(16) modules(0) v1AttrCert(15) } ;

   -- Cryptographic Message Syntax


   CONTENT-TYPE ::= TYPE-IDENTIFIER
   ContentType ::= OBJECT IDENTIFIER

   ContentInfo ::= SEQUENCE {
       contentType        CONTENT-TYPE.
                       &id({ContentSet}),
```

```
        content            [0] EXPLICIT CONTENT-TYPE.
                            &Type({ContentSet}{@contentType})}

    ContentSet CONTENT-TYPE ::= {
        --  Define the set of content types to be recognized.
        ct-Data | ct-SignedData | ct-EncryptedData | ct-EnvelopedData |
        ct-AuthenticatedData | ct-DigestedData, ... }

    SignedData ::= SEQUENCE {
        version CMSVersion,
        digestAlgorithms SET OF DigestAlgorithmIdentifier,
        encapContentInfo EncapsulatedContentInfo,
        certificates [0] IMPLICIT CertificateSet OPTIONAL,
        crls [1] IMPLICIT RevocationInfoChoices OPTIONAL,
        signerInfos SignerInfos }

    DigestAlgorithmList ALGORITHM ::= { -- alg-sha-1 | alg-md5, -- ... }

    SignatureAlgorithmList ALGORITHM ::=
        { -- alg-dsa-with-sha1 | alg-md5WithRSAEncryption --
            -- | alg-sha1WithRSAEncryption, -- ... }

    SignerInfos ::= SET OF SignerInfo

    EncapsulatedContentInfo ::= SEQUENCE {
        eContentType       CONTENT-TYPE.&id({ContentSet}),
        eContent           [0] EXPLICIT OCTET STRING
                ( CONTAINING CONTENT-TYPE.
                    &Type({ContentSet}{@eContentType})) OPTIONAL }

    SignerInfo ::= SEQUENCE {
        version CMSVersion,
        sid SignerIdentifier,
        digestAlgorithm DigestAlgorithmIdentifier,
        signedAttrs [0] IMPLICIT AuthAttributes OPTIONAL,
        signatureAlgorithm SignatureAlgorithmIdentifier,
        signature SignatureValue,
        unsignedAttrs [1] IMPLICIT Attributes
            {{UnsignedAttributes}} OPTIONAL }

    AuthAttributes ::= Attributes {{ SignedAttributes }}

    SignerIdentifier ::= CHOICE {
        issuerAndSerialNumber IssuerAndSerialNumber,
        subjectKeyIdentifier [0] SubjectKeyIdentifier }

    SignedAttributes ATTRIBUTE ::=
        { attr-signingTime | attr-messageDigest | attr-contentType, ... }
```

```
   UnsignedAttributes ATTRIBUTE ::= { attr-countersignature, ... }

   SignatureValue ::= OCTET STRING

   EnvelopedData ::= SEQUENCE {
       version CMSVersion,
       originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
       recipientInfos RecipientInfos,
       encryptedContentInfo EncryptedContentInfo,
       unprotectedAttrs [1] IMPLICIT Attributes
           {{UnprotectedAttributes}} OPTIONAL }

   OriginatorInfo ::= SEQUENCE {
       certs [0] IMPLICIT CertificateSet OPTIONAL,
       crls [1] IMPLICIT RevocationInfoChoices OPTIONAL }

   RecipientInfos ::= SET SIZE (1..MAX) OF RecipientInfo

   EncryptedContentInfo ::= SEQUENCE {
       contentType        CONTENT-TYPE.&id({ContentSet}),
       contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
       encryptedContent   [0] IMPLICIT OCTET STRING OPTIONAL }

   -- If you want to do constraints, you might use:
   -- EncryptedContentInfo ::= SEQUENCE {
   --  contentType        CONTENT-TYPE.&id({ContentSet}),
   --  contentEncryptionAlgorithm ContentEncryptionAlgorithmIdentifier,
   --  encryptedContent   [0] IMPLICIT ENCRYPTED {CONTENT-TYPE.
   --      &Type({ContentSet}{@contentType}) OPTIONAL }
   -- ENCRYPTED {ToBeEncrypted} ::= OCTET STRING ( CONSTRAINED BY
   --        { ToBeEncrypted } )

   ContentEncryptionAlgorithmList ALGORITHM ::=
       { -- alg-des-ede3-cbc | alg-rd2-cbc, -- ... }

   UnprotectedAttributes ATTRIBUTE ::=  { ... }

   RecipientInfo ::= CHOICE {
       ktri       KeyTransRecipientInfo,
       kari   [1] KeyAgreeRecipientInfo,
       kekri  [2] KEKRecipientInfo,
       pwri   [3] PasswordRecipientInfo,
       ori    [4] OtherRecipientInfo }

   EncryptedKey ::= OCTET STRING

   KeyTransRecipientInfo ::= SEQUENCE {
       version CMSVersion,  -- always set to 0 or 2
```

```
        rid RecipientIdentifier,
        keyEncryptionAlgorithm AlgorithmIdentifier
            {{KeyTransportAlgorithmList}},
        encryptedKey EncryptedKey }

    KeyTransportAlgorithmList ALGORITHM ::=
        { -- alg-rsaEncryption, -- ... }

    RecipientIdentifier ::= CHOICE {
        issuerAndSerialNumber IssuerAndSerialNumber,
        subjectKeyIdentifier [0] SubjectKeyIdentifier }

    KeyAgreeRecipientInfo ::= SEQUENCE {
        version CMSVersion,  -- always set to 3
        originator [0] EXPLICIT OriginatorIdentifierOrKey,
        ukm [1] EXPLICIT UserKeyingMaterial OPTIONAL,
        keyEncryptionAlgorithm AlgorithmIdentifier
            {{KeyAgreementAlgorithmList}},
        recipientEncryptedKeys RecipientEncryptedKeys }

    KeyAgreementAlgorithmList ALGORITHM ::=
        { -- alg-ESDH | alg-SSDH, -- ... }

    OriginatorIdentifierOrKey ::= CHOICE {
        issuerAndSerialNumber IssuerAndSerialNumber,
        subjectKeyIdentifier [0] SubjectKeyIdentifier,
        originatorKey [1] OriginatorPublicKey }

    OriginatorPublicKey ::= SEQUENCE {
        algorithm AlgorithmIdentifier {{AlgorithmList}},
        publicKey BIT STRING }

    RecipientEncryptedKeys ::= SEQUENCE OF RecipientEncryptedKey

    RecipientEncryptedKey ::= SEQUENCE {
        rid KeyAgreeRecipientIdentifier,
        encryptedKey EncryptedKey }

    KeyEncryptKeyAlgorithmList ALGORITHM ::=
        { -- alg-CMS3DESwrap | alg-CMSRC2wrap, -- ... }

    KeyEncryptionAlgorithmList ALGORITHM ::= { ... }

    KeyAgreeRecipientIdentifier ::= CHOICE {
        issuerAndSerialNumber IssuerAndSerialNumber,
        rKeyId [0] IMPLICIT RecipientKeyIdentifier }

    RecipientKeyIdentifier ::= SEQUENCE {
```

```
        subjectKeyIdentifier SubjectKeyIdentifier,
        date GeneralizedTime OPTIONAL,
        other OtherKeyAttribute OPTIONAL }

    SubjectKeyIdentifier ::= OCTET STRING

    KEKRecipientInfo ::= SEQUENCE {
        version CMSVersion,  -- always set to 4
        kekid KEKIdentifier,
        keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
        encryptedKey EncryptedKey }

    KEKIdentifier ::= SEQUENCE {
        keyIdentifier OCTET STRING,
        date GeneralizedTime OPTIONAL,
        other OtherKeyAttribute OPTIONAL }

    PasswordRecipientInfo ::= SEQUENCE {
        version CMSVersion,   -- always set to 0
        keyDerivationAlgorithm [0] KeyDerivationAlgorithmIdentifier
                                OPTIONAL,
        keyEncryptionAlgorithm KeyEncryptionAlgorithmIdentifier,
        encryptedKey EncryptedKey }

    OTHER-RECIPIENT ::= TYPE-IDENTIFIER

    OtherRecipientInfo ::= SEQUENCE {
        oriType    OTHER-RECIPIENT.
                &id({SupportedOtherRecipInfo}),
        oriValue   OTHER-RECIPIENT.
                &Type({SupportedOtherRecipInfo}{@oriType})}

    SupportedOtherRecipInfo OTHER-RECIPIENT ::= { ... }

    DigestedData ::= SEQUENCE {
        version CMSVersion,
        digestAlgorithm DigestAlgorithmIdentifier,
        encapContentInfo EncapsulatedContentInfo,
        digest Digest }

    Digest ::= OCTET STRING

    EncryptedData ::= SEQUENCE {
        version CMSVersion,
        encryptedContentInfo EncryptedContentInfo,
        unprotectedAttrs [1] IMPLICIT Attributes
            {{UnprotectedAttributes}} OPTIONAL }
```

```
   AuthenticatedData ::= SEQUENCE {
       version CMSVersion,
       originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
       recipientInfos RecipientInfos,
       macAlgorithm MessageAuthenticationCodeAlgorithm,
       digestAlgorithm [1] DigestAlgorithmIdentifier OPTIONAL,
       encapContentInfo EncapsulatedContentInfo,
       authAttrs [2] IMPLICIT AuthAttributes OPTIONAL,
       mac MessageAuthenticationCode,
       unauthAttrs [3] IMPLICIT UnauthAttributes OPTIONAL }

   AuthAttributes ::= SET SIZE (1..MAX) OF Attribute
       {{SupportedAttributes}}

   UnauthAttributes ::= SET SIZE (1..MAX) OF Attribute
       {{SupportedAttributes}}

   MessageAuthenticationCode ::= OCTET STRING

   DigestAlgorithmIdentifier ::= AlgorithmIdentifier
       {{DigestAlgorithmList}}

   SignatureAlgorithmIdentifier ::= AlgorithmIdentifier
       {{SignatureAlgorithmList}}

   KeyEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
       {{KeyEncryptionAlgorithmList}}

   ContentEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
       {{ContentEncryptionAlgorithmList}}

   MessageAuthenticationCodeAlgorithm ::= AlgorithmIdentifier
       {{AlgorithmList}}

   KeyDerivationAlgorithmIdentifier ::= AlgorithmIdentifier
       {{AlgorithmList}}

   AlgorithmList ALGORITHM ::= { ... }

   RevocationInfoChoices ::= SET OF RevocationInfoChoice

   RevocationInfoChoice ::= CHOICE {
       crl CertificateList,
       other [1] IMPLICIT OtherRevocationInfoFormat }

   OTHER-REVOK-INFO ::= TYPE-IDENTIFIER

   OtherRevocationInfoFormat ::= SEQUENCE {
```

```
        otherRevInfoFormat    OTHER-REVOK-INFO.
                &id({SupportedOtherRevokInfo}),
        otherRevInfo          OTHER-REVOK-INFO.
                &Type({SupportedOtherRevokInfo}{@otherRevInfoFormat})}

    SupportedOtherRevokInfo OTHER-REVOK-INFO ::= { ... }

    CertificateChoices ::= CHOICE {
        certificate Certificate,
        extendedCertificate [0] IMPLICIT ExtendedCertificate, -- Obsolete
        v1AttrCert [1] IMPLICIT AttributeCertificateV1,       -- Obsolete
        v2AttrCert [2] IMPLICIT AttributeCertificateV2,
        other [3] IMPLICIT OtherCertificateFormat }

    AttributeCertificateV2 ::= AttributeCertificate

    OTHER-CERT-FMT ::= TYPE-IDENTIFIER

    OtherCertificateFormat ::= SEQUENCE {
        otherCertFormat OTHER-CERT-FMT.
                &id({SupportedCertFormats}),
        otherCert       OTHER-CERT-FMT.
                &Type({SupportedCertFormats}{@otherCertFormat})}

    SupportedCertFormats OTHER-CERT-FMT ::= { ... }

    CertificateSet ::= SET OF CertificateChoices

    IssuerAndSerialNumber ::= SEQUENCE {
        issuer Name,
        serialNumber CertificateSerialNumber }

    CMSVersion ::= INTEGER  { v0(0), v1(1), v2(2), v3(3), v4(4), v5(5) }

    UserKeyingMaterial ::= OCTET STRING

    KEY-ATTRIBUTE ::= TYPE-IDENTIFIER

    OtherKeyAttribute ::= SEQUENCE {
        keyAttrId  KEY-ATTRIBUTE.
                &id({SupportedKeyAttributes}),
        keyAttr    KEY-ATTRIBUTE.
                &Type({SupportedKeyAttributes}{@keyAttrId})}

    SupportedKeyAttributes KEY-ATTRIBUTE ::= { ... }

    -- Content Type Object Identifiers
```

```
   id-ct-contentInfo OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) ct(1) 6 }

   ct-Data CONTENT-TYPE ::= {OCTET STRING IDENTIFIED BY id-data}

   id-data OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs7(7) 1 }

   ct-SignedData CONTENT-TYPE ::=
       { SignedData IDENTIFIED BY id-signedData}

   id-signedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }

   ct-EnvelopedData CONTENT-TYPE ::=
       { EnvelopedData IDENTIFIED BY id-envelopedData}

   id-envelopedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs7(7) 3 }

   ct-DigestedData CONTENT-TYPE ::=
       { DigestedData IDENTIFIED BY id-digestedData}

   id-digestedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs7(7) 5 }

   ct-EncryptedData CONTENT-TYPE ::=
       { EncryptedData IDENTIFIED BY id-encryptedData}

   id-encryptedData OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs7(7) 6 }

   ct-AuthenticatedData CONTENT-TYPE ::=
       { AuthenticatedData IDENTIFIED BY id-ct-authData}

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

   -- The CMS Attributes

   MessageDigest ::= OCTET STRING

   SigningTime  ::= Time

   Time ::= CHOICE {
       utcTime UTCTime,
       generalTime GeneralizedTime }
```

```
   Countersignature ::= SignerInfo

   -- Attribute Object Identifiers

   attr-contentType ATTRIBUTE ::=
       { ContentType IDENTIFIED BY id-contentType }

   id-contentType OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs9(9) 3 }

   attr-messageDigest ATTRIBUTE ::=
       { OCTET STRING IDENTIFIED BY id-messageDigest}

   id-messageDigest OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs9(9) 4 }

   attr-signingTime ATTRIBUTE ::=
       { Time IDENTIFIED BY id-signingTime }

   id-signingTime OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs9(9) 5 }

   attr-countersignature ATTRIBUTE ::=
       { SignerInfo IDENTIFIED BY id-countersignature }

   id-countersignature OBJECT IDENTIFIER ::= { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs9(9) 6 }

   -- Obsolete Extended Certificate syntax from PKCS#6

   ExtendedCertificateOrCertificate ::= CHOICE {
       certificate Certificate,
       extendedCertificate [0] IMPLICIT ExtendedCertificate }

   ExtendedCertificate ::= SEQUENCE {
       extendedCertificateInfo ExtendedCertificateInfo,
       signatureAlgorithm SignatureAlgorithmIdentifier,
       signature Signature }

   ExtendedCertificateInfo ::= SEQUENCE {
       version CMSVersion,
       certificate Certificate,
       attributes UnauthAttributes }

   Signature ::= BIT STRING

   --  Class definitions used in the module
```

```
   AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
       algorithm  ALGORITHM.&id({IOSet}),
       parameters ALGORITHM.&Type({IOSet}{@algorithm}) OPTIONAL }

   ATTRIBUTE ::= TYPE-IDENTIFIER

   Attribute{ATTRIBUTE: AttrList} ::= SEQUENCE {
       attrType           ATTRIBUTE.
                  &id({AttrList}),
       attrValues         SET OF ATTRIBUTE.
                  &Type({AttrList}{@attrType})  }

   SupportedAttributes ATTRIBUTE ::= { ... }

   Attributes { ATTRIBUTE:AttrList } ::=
       SET SIZE (1..MAX) OF Attribute {{ AttrList }}

   END
```

5. ASN.1 Module for RFC 4108

```
   CMSFirmwareWrapper
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
       smime(16) modules(0) cms-firmware-wrap(22) }
   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN

   IMPORTS

   EnvelopedData
   FROM CryptographicMessageSyntax
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
       smime(16) modules(0) cms-2004(24) };


   -- Firmware Package Content Type and Object Identifier

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

   FirmwarePkgData ::= OCTET STRING

   -- Firmware Package Signed Attributes and Object Identifiers

   id-aa-firmwarePackageID OBJECT IDENTIFIER ::= {
       iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
```

```
        smime(16) aa(2) 35 }

    FirmwarePackageIdentifier ::= SEQUENCE {
        name PreferredOrLegacyPackageIdentifier,
        stale PreferredOrLegacyStalePackageIdentifier OPTIONAL }

    PreferredOrLegacyPackageIdentifier ::= CHOICE {
        preferred PreferredPackageIdentifier,
        legacy OCTET STRING }

    PreferredPackageIdentifier ::= SEQUENCE {
        fwPkgID OBJECT IDENTIFIER,
        verNum INTEGER (0..MAX) }

    PreferredOrLegacyStalePackageIdentifier ::= CHOICE {
        preferredStaleVerNum INTEGER (0..MAX),
        legacyStaleVersion OCTET STRING }


    id-aa-targetHardwareIDs OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
        smime(16) aa(2) 36 }

    TargetHardwareIdentifiers ::= SEQUENCE OF OBJECT IDENTIFIER


    id-aa-decryptKeyID OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
        smime(16) aa(2) 37 }

    DecryptKeyIdentifier ::= OCTET STRING

    id-aa-implCryptoAlgs OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
        smime(16) aa(2) 38 }

    ImplementedCryptoAlgorithms ::= SEQUENCE OF OBJECT IDENTIFIER

    id-aa-implCompressAlgs OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
        smime(16) aa(2) 43 }

    ImplementedCompressAlgorithms ::= SEQUENCE OF OBJECT IDENTIFIER

    id-aa-communityIdentifiers OBJECT IDENTIFIER ::= {
        iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
        smime(16) aa(2) 40 }
```

```
   CommunityIdentifiers ::= SEQUENCE OF CommunityIdentifier

   CommunityIdentifier ::= CHOICE {
       communityOID OBJECT IDENTIFIER,
       hwModuleList HardwareModules }

   HardwareModules ::= SEQUENCE {
       hwType OBJECT IDENTIFIER,
       hwSerialEntries SEQUENCE OF HardwareSerialEntry }

   HardwareSerialEntry ::= CHOICE {
       all NULL,
       single OCTET STRING,
       block SEQUENCE {
       low OCTET STRING,
       high OCTET STRING } }


   id-aa-firmwarePackageInfo OBJECT IDENTIFIER ::= {
       iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
       smime(16) aa(2) 42 }

   FirmwarePackageInfo ::= SEQUENCE {
       fwPkgType INTEGER OPTIONAL,
       dependencies SEQUENCE OF
       PreferredOrLegacyPackageIdentifier OPTIONAL }

   -- Firmware Package Unsigned Attributes and Object Identifiers

   id-aa-wrappedFirmwareKey OBJECT IDENTIFIER ::= {
       iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
       smime(16) aa(2) 39 }

   WrappedFirmwareKey ::= EnvelopedData

   -- Firmware Package Load Receipt Content Type and Object Identifier

   id-ct-firmwareLoadReceipt OBJECT IDENTIFIER ::= {
       iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
       smime(16) ct(1) 17 }

   FirmwarePackageLoadReceipt ::= SEQUENCE {
       version FWReceiptVersion DEFAULT v1,
       hwType OBJECT IDENTIFIER,
       hwSerialNum OCTET STRING,
       fwPkgName PreferredOrLegacyPackageIdentifier,
       trustAnchorKeyID OCTET STRING OPTIONAL,
       decryptKeyID [1] OCTET STRING OPTIONAL }
```

```
   FWReceiptVersion ::= INTEGER { v1(1) }

   -- Firmware Package Load Error Report Content Type
   -- and Object Identifier

   id-ct-firmwareLoadError OBJECT IDENTIFIER ::= {
       iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
       smime(16) ct(1) 18 }

   FirmwarePackageLoadError ::= SEQUENCE {
       version FWErrorVersion DEFAULT v1,
       hwType OBJECT IDENTIFIER,
       hwSerialNum OCTET STRING,
       errorCode FirmwarePackageLoadErrorCode,
       vendorErrorCode VendorLoadErrorCode OPTIONAL,
       fwPkgName PreferredOrLegacyPackageIdentifier OPTIONAL,
       config [1] SEQUENCE OF CurrentFWConfig OPTIONAL }

   FWErrorVersion ::= INTEGER { v1(1) }

   CurrentFWConfig ::= SEQUENCE {
       fwPkgType INTEGER OPTIONAL,
       fwPkgName PreferredOrLegacyPackageIdentifier }

   FirmwarePackageLoadErrorCode ::= ENUMERATED {
       decodeFailure                (1),
       badContentInfo               (2),
       badSignedData                (3),
       badEncapContent              (4),
       badCertificate               (5),
       badSignerInfo                (6),
       badSignedAttrs               (7),
       badUnsignedAttrs             (8),
       missingContent               (9),
       noTrustAnchor               (10),
       notAuthorized               (11),
       badDigestAlgorithm          (12),
       badSignatureAlgorithm       (13),
       unsupportedKeySize          (14),
       signatureFailure            (15),
       contentTypeMismatch         (16),
       badEncryptedData            (17),
       unprotectedAttrsPresent     (18),
       badEncryptContent           (19),
       badEncryptAlgorithm         (20),
       missingCiphertext           (21),
       noDecryptKey                (22),
       decryptFailure              (23),
```

```
        badCompressAlgorithm        (24),
        missingCompressedContent    (25),
        decompressFailure           (26),
        wrongHardware               (27),
        stalePackage                (28),
        notInCommunity              (29),
        unsupportedPackageType      (30),
        missingDependency           (31),
        wrongDependencyVersion      (32),
        insufficientMemory          (33),
        badFirmware                 (34),
        unsupportedParameters       (35),
        breaksDependency            (36),
        otherError                  (99) }

    VendorLoadErrorCode ::= INTEGER

    -- Other Name syntax for Hardware Module Name

    id-on-hardwareModuleName OBJECT IDENTIFIER ::= {
        iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) pkix(7) on(8) 4 }

    HardwareModuleName ::= SEQUENCE {
        hwType OBJECT IDENTIFIER,
        hwSerialNum OCTET STRING }

    END
```

6. ASN.1 Module for RFC 5035

```
  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

  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) }

  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) }

  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
  -- tp have at least one entry.  MAX indicates the upper bound is
  -- unspecified.  Implementations are free to choose an upper bound
  -- that suits their environment.

  -- 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

  SECURITY-CATEGORY ::= TYPE-IDENTIFIER

  SecurityCategory ::= SEQUENCE {
      type  [0] SECURITY-CATEGORY.
              &id({SupportedSecurityCategories}),
      value [1] SECURITY-CATEGORY.
              &Type({SupportedSecurityCategories}{@type})
  }

  SupportedSecurityCategories SECURITY-CATEGORY ::= { ... }

  --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
```

7. ASN.1 Module for RFC 5083

```
CMS-AuthEnvelopedData-2007
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
    pkcs-9(9) smime(16) modules(0) cms-authEnvelopedData(31) }
DEFINITIONS IMPLICIT TAGS ::=
BEGIN

IMPORTS

AuthAttributes, CMSVersion, EncryptedContentInfo,
    MessageAuthenticationCode, OriginatorInfo, RecipientInfos,
    UnauthAttributes
FROM CryptographicMessageSyntax2004
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) modules(0) cms-2004(24) } ;

id-ct-authEnvelopedData OBJECT IDENTIFIER ::=
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) ct(1) 23 }

AuthEnvelopedData ::= SEQUENCE {
    version CMSVersion,
    originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
    recipientInfos RecipientInfos,
    authEncryptedContentInfo EncryptedContentInfo,
    authAttrs [1] IMPLICIT AuthAttributes OPTIONAL,
    mac MessageAuthenticationCode,
    unauthAttrs [2] IMPLICIT UnauthAttributes OPTIONAL }

END
```

8. ASN.1 Module for RFC 5084

```
CMS-AES-CCM-and-AES-GCM
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
    pkcs-9(9) smime(16) modules(0) cms-aes-ccm-and-gcm(32) }
DEFINITIONS IMPLICIT TAGS ::=
BEGIN

-- Object Identifiers

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

id-aes128-CCM OBJECT IDENTIFIER ::= { aes 7 }

id-aes192-CCM OBJECT IDENTIFIER ::= { aes 27 }

id-aes256-CCM OBJECT IDENTIFIER ::= { aes 47 }

id-aes128-GCM OBJECT IDENTIFIER ::= { aes 6 }

id-aes192-GCM OBJECT IDENTIFIER ::= { aes 26 }

id-aes256-GCM OBJECT IDENTIFIER ::= { aes 46 }

-- Parameters for AigorithmIdentifier

CCMParameters ::= SEQUENCE {
    aes-nonce         OCTET STRING (SIZE(7..13)),
    aes-ICVlen        AES-CCM-ICVlen DEFAULT 12 }

AES-CCM-ICVlen ::= INTEGER (4 | 6 | 8 | 10 | 12 | 14 | 16)

GCMParameters ::= SEQUENCE {
    aes-nonce        OCTET STRING, -- recommended size is 12 octets
    aes-ICVlen       AES-GCM-ICVlen DEFAULT 12 }

AES-GCM-ICVlen ::= INTEGER (12 | 13 | 14 | 15 | 16)

END
```

9. Security Considerations

Even though all the RFCs in this document are security-related, the document itself does not have any security considerations. The ASN.1 modules keep the same bits-on-the-wire as the modules that they replace.

10. Normative References

[ASN1-2002] ITU-T, "ITU-T Recommendation X.680 Information technology [ETH] Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T X.680, 2002.

[NEW-PKIX] Hoffman, P. and J. Schaad, "New ASN.1 Modules for PKIX", draft-hoffman-pkix-new-asn1 (work in progress), November 2007.

[RFC3370] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, August 2002.

[RFC3851] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

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

[RFC4108] Housley, R., "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages", RFC 4108, August 2005.

[RFC5035] Schaad, J., "Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility", RFC 5035, August 2007.

[RFC5083] Housley, R., "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type", RFC-to-be 5083, November 2007.

[RFC5084] Housley, R., "Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS)", RFC-to-be 5084, November 2007.

Authors' Addresses

Paul Hoffman
VPN Consortium
127 Segre Place
Santa Cruz, CA 95060
US

Phone: 1-831-426-9827
Email: paul.hoffman@vpnc.org

Jim Schaad
Soaring Hawk Consulting

Email: jimsch@exmsft.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).