Internet Engineering Task Force (IETF)
Request for Comments: 5911
Category: Informational
ISSN: 2070-1721

P. Hoffman
VPN Consortium
J. Schaad
Soaring Hawk Consulting

June 2010

New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME

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.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5911.

Copyright Notice

Table of Contents

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 complete.

- RFC 3370, CMS Algorithms [RFC3370]
- RFC 3565, Use of AES in CMS [RFC3565]
- 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 4998, Evidence Record Syntax (ERS) [RFC4998]
- RFC 5035, Enhanced Security Services (ESS) [RFC5035]
- RFC 5083, CMS Authenticated-Enveloped-Data Content Type [RFC5083]
- RFC 5084, Using AES-CCM and AES-GCM Authenticated Encryption in CMS [RFC5084]
- RFC 5275, CMS Symmetric Key Management and Distribution [RFC5275]

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 [RFC5912] can stand on their own and do not need to import definitions from anywhere else. Also note that the ASN.1 modules in this document have references in their text comments that need to be looked up in original RFCs, and that some of those references may have already been superseded by later RFCs.

The document also includes a module of common definitions called "AlgorithmInformation". These definitions are used here and in [RFC5912].

Note that some of the modules here import definitions from the common definitions module, "PKIX-CommonTypes", in [RFC5912].

1.1. Design Notes

The modules in this document use the object model available in the 2002 ASN.1 documents to a great extent. Objects for each of the different algorithm types are defined. Also, all of the places where the 1988 ASN.1 syntax had ANY holes to allow for variable syntax now use objects.

Much like the way that the PKIX and S/MIME working groups use the prefix of id- for object identifiers, this document has also adopted a set of two-, three-, and four-letter prefixes to allow for quick identification of the type of an object based on its name. This allows, for example, the same back half of the name to be used for the different objects. Thus, "id-sha1" is the object identifier, while "mda-sha1" is the message digest object for "sha1".

One or more object sets for the different types of algorithms are defined. A single consistent name for each different algorithm type is used. For example, an object set named PublicKeys contains the public keys defined in that module. If no public keys are defined, then the object set is not created. When importing these object sets into an ASN.1 module, one needs to be able to distinguish between the different object sets with the same name. This is done by using both the module name (as specified in the IMPORT statement) and the object set name. For example, in the module for RFC 5280:

```
PublicKeys FROM PKIXAlgs-2008 { 1 3 6 1 5 5 7 0 995 }
PublicKeys FROM PKIX1-PSS-OAEP-Algorithms { 1 3 6 1 5 5 7  33 }

PublicKeyAlgorithms PUBLIC-KEY ::= { PKIXAlgs-2008.PublicKeys, ...,
    PKIX1-PSS-OAEP-Algorithms.PublicKeys }
```

2. ASN.1 Module AlgorithmInformation

This section contains a module that is imported by many other modules in this document. Note that this module is also given in [RFC5912]. This module does not come from any existing RFC.

```
AlgorithmInformation-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58)}
```

```
DEFINITIONS EXPLICIT TAGS ::=
BEGIN
EXPORTS ALL;
IMPORTS

KeyUsage
FROM PKIX1Implicit-2009
    {iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-pkix1-implicit-02(59)} ;

--  Suggested prefixes for algorithm objects are:
--
--  mda-   Message Digest Algorithms
--  sa-    Signature Algorithms
--  kta-   Key Transport Algorithms (Asymmetric)
--  kaa-   Key Agreement Algorithms  (Asymmetric)
--  kwa-   Key Wrap Algorithms (Symmetric)
--  kda-   Key Derivation Algorithms
--  maca-  Message Authentication Code Algorithms
--  pk-    Public Key
--  cea-   Content (symmetric) Encryption Algorithms
--  cap-   S/MIME Capabilities

ParamOptions ::= ENUMERATED {
   required,         -- Parameters MUST be encoded in structure
   preferredPresent, -- Parameters SHOULD be encoded in structure
   preferredAbsent,  -- Parameters SHOULD NOT be encoded in structure
   absent,           -- Parameters MUST NOT be encoded in structure
   inheritable,      -- Parameters are inherited if not present
   optional,         -- Parameters MAY be encoded in the structure
   ...
}

--  DIGEST-ALGORITHM
--
--  Describes the basic information for ASN.1 and a digest
--      algorithm.
--
--  &id - contains the OID identifying the digest algorithm
--  &Params - if present, contains the type for the algorithm
--               parameters; if absent, implies no parameters
--  &paramPresence - parameter presence requirement
--
--  Additional information such as the length of the hash could have
--      been encoded.  Without a clear understanding of what information
--      is needed by applications, such extraneous information was not
--      considered to be of sufficient importance.
```

```
--
--  Example:
--  mda-sha1 DIGEST-ALGORITHM ::= {
--      IDENTIFIER id-sha1
--      PARAMS TYPE NULL ARE preferredAbsent
--  }

DIGEST-ALGORITHM ::= CLASS {
    &id                 OBJECT IDENTIFIER UNIQUE,
    &Params             OPTIONAL,
    &paramPresence      ParamOptions DEFAULT absent
} WITH SYNTAX {
    IDENTIFIER &id
    [PARAMS [TYPE &Params] ARE &paramPresence ]
}

--  SIGNATURE-ALGORITHM
--
--  Describes the basic properties of a signature algorithm
--
--  &id - contains the OID identifying the signature algorithm
--  &Value - contains a type definition for the value structure of
--              the signature; if absent, implies that no ASN.1
--              encoding is performed on the value
--  &Params - if present, contains the type for the algorithm
--               parameters; if absent, implies no parameters
--  &paramPresence - parameter presence requirement
--  &HashSet - The set of hash algorithms used with this
--                  signature algorithm
--  &PublicKeySet - the set of public key algorithms for this
--                  signature algorithm
--  &smimeCaps - contains the object describing how the S/MIME
--              capabilities are presented.
--
--  Example:
--  sig-RSA-PSS SIGNATURE-ALGORITHM ::= {
--     IDENTIFIER id-RSASSA-PSS
--     PARAMS TYPE RSASSA-PSS-params ARE required
--     HASHES { mda-sha1 | mda-md5, ... }
--     PUBLIC-KEYS { pk-rsa | pk-rsa-pss }
-- }

SIGNATURE-ALGORITHM ::= CLASS {
    &id             OBJECT IDENTIFIER UNIQUE,
    &Value          OPTIONAL,
    &Params         OPTIONAL,
    &paramPresence  ParamOptions DEFAULT absent,
    &HashSet        DIGEST-ALGORITHM OPTIONAL,
    &PublicKeySet   PUBLIC-KEY OPTIONAL,
    &smimeCaps      SMIME-CAPS OPTIONAL
} WITH SYNTAX {
    IDENTIFIER &id
    [VALUE &Value]
    [PARAMS [TYPE &Params] ARE &paramPresence ]
    [HASHES &HashSet]
    [PUBLIC-KEYS &PublicKeySet]
    [SMIME-CAPS &smimeCaps]
}

--  PUBLIC-KEY
--
--  Describes the basic properties of a public key
--
--  &id - contains the OID identifying the public key
--  &KeyValue - contains the type for the key value
--  &Params - if present, contains the type for the algorithm
--                parameters; if absent, implies no parameters
--  &paramPresence - parameter presence requirement
--  &keyUsage - contains the set of bits that are legal for this
--              key type.  Note that it does not make any statement
--              about how bits may be paired.
--  &PrivateKey - contains a type structure for encoding the private
--              key information.
--
--  Example:
--  pk-rsa-pss PUBLIC-KEY ::= {
--      IDENTIFIER id-RSASSA-PSS
--      KEY RSAPublicKey
--      PARAMS TYPE RSASSA-PSS-params ARE optional
--      CERT-KEY-USAGE { .... }
--  }

PUBLIC-KEY ::= CLASS {
    &id             OBJECT IDENTIFIER UNIQUE,
    &KeyValue       OPTIONAL,
    &Params         OPTIONAL,
    &paramPresence  ParamOptions DEFAULT absent,
    &keyUsage       KeyUsage OPTIONAL,
    &PrivateKey     OPTIONAL
} WITH SYNTAX {
    IDENTIFIER &id
    [KEY &KeyValue]
    [PARAMS [TYPE &Params] ARE &paramPresence]
    [CERT-KEY-USAGE &keyUsage]
    [PRIVATE-KEY &PrivateKey]
}
```

```
--  KEY-TRANSPORT
--
--  Describes the basic properties of a key transport algorithm
--
--  &id - contains the OID identifying the key transport algorithm
--  &Params - if present, contains the type for the algorithm
--                parameters; if absent, implies no parameters
--  &paramPresence - parameter presence requirement
--  &PublicKeySet - specifies which public keys are used with
--                        this algorithm
--  &smimeCaps - contains the object describing how the S/MIME
--               capabilities are presented.
--
--  Example:
--  kta-rsaTransport KEY-TRANSPORT ::= {
--      IDENTIFIER &id
--      PARAMS TYPE NULL ARE required
--      PUBLIC-KEYS  { pk-rsa | pk-rsa-pss }
--  }

KEY-TRANSPORT ::= CLASS {
    &id                 OBJECT IDENTIFIER UNIQUE,
    &Params             OPTIONAL,
    &paramPresence      ParamOptions DEFAULT absent,
    &PublicKeySet       PUBLIC-KEY OPTIONAL,
    &smimeCaps          SMIME-CAPS OPTIONAL
} WITH SYNTAX {
    IDENTIFIER &id
    [PARAMS [TYPE &Params] ARE &paramPresence]
    [PUBLIC-KEYS &PublicKeySet]
    [SMIME-CAPS &smimeCaps]
}

--  KEY-AGREE
--
--  Describes the basic properties of a key agreement algorithm
--
--  &id - contains the OID identifying the key agreement algorithm
--  &Params - if present, contains the type for the algorithm
--                parameters; if absent, implies no parameters
--  &paramPresence - parameter presence requirement
--  &PublicKeySet - specifies which public keys are used with
--                         this algorithm
--  &Ukm - type of user keying material used
--  &ukmPresence - specifies the requirements to define the UKM field
--  &smimeCaps - contains the object describing how the S/MIME
--               capabilities are presented.
--
```

```
--  Example:
--  kaa-dh-static-ephemeral KEY-AGREE ::= {
--      IDENTIFIER id-alg-ESDH
--      PARAMS TYPE KeyWrapAlgorithm ARE required
--      PUBLIC-KEYS {
--         {IDENTIFIER dh-public-number KEY DHPublicKey
--            PARAMS TYPE DHDomainParameters ARE inheritable }
--      }
--      - - UKM should be present but is not separately ASN.1-encoded
--      UKM ARE preferredPresent
--  }

KEY-AGREE ::= CLASS {
    &id             OBJECT IDENTIFIER UNIQUE,
    &Params         OPTIONAL,
    &paramPresence  ParamOptions DEFAULT absent,
    &PublicKeySet   PUBLIC-KEY OPTIONAL,
    &Ukm            OPTIONAL,
    &ukmPresence    ParamOptions DEFAULT absent,
    &smimeCaps      SMIME-CAPS OPTIONAL
} WITH SYNTAX {
    IDENTIFIER &id
    [PARAMS [TYPE &Params] ARE &paramPresence]
    [PUBLIC-KEYS &PublicKeySet]
    [UKM [TYPE &Ukm] ARE &ukmPresence]
    [SMIME-CAPS &smimeCaps]
}

--  KEY-WRAP
--
--  Describes the basic properties of a key wrap algorithm
--
--  &id - contains the OID identifying the key wrap algorithm
--  &Params - if present, contains the type for the algorithm
--               parameters; if absent, implies no parameters
--  &paramPresence - parameter presence requirement
--  &smimeCaps - contains the object describing how the S/MIME
--              capabilities are presented.
--
--  Example:
--  kwa-cms3DESwrap KEY-WRAP ::= {
--      IDENTIFIER id-alg-CMS3DESwrap
--      PARAMS TYPE NULL ARE required
--  }

KEY-WRAP ::= CLASS {
    &id                 OBJECT IDENTIFIER UNIQUE,
    &Params             OPTIONAL,
    &paramPresence     ParamOptions DEFAULT absent,
    &smimeCaps         SMIME-CAPS OPTIONAL
} WITH SYNTAX {
    IDENTIFIER &id
    [PARAMS [TYPE &Params] ARE &paramPresence]
    [SMIME-CAPS &smimeCaps]
}

--  KEY-DERIVATION
--
--  Describes the basic properties of a key derivation algorithm
--
--  &id - contains the OID identifying the key derivation algorithm
--  &Params - if present, contains the type for the algorithm
--               parameters; if absent, implies no parameters
--  &paramPresence - parameter presence requirement
--  &smimeCaps - contains the object describing how the S/MIME
--              capabilities are presented.
--
--  Example:
--  kda-pbkdf2 KEY-DERIVATION ::= {
--      IDENTIFIER id-PBKDF2
--      PARAMS TYPE PBKDF2-params ARE required
--  }

KEY-DERIVATION ::= CLASS {
    &id                OBJECT IDENTIFIER UNIQUE,
    &Params            OPTIONAL,
    &paramPresence     ParamOptions DEFAULT absent,
    &smimeCaps         SMIME-CAPS OPTIONAL
} WITH SYNTAX {
    IDENTIFIER &id
    [PARAMS [TYPE &Params] ARE &paramPresence]
    [SMIME-CAPS &smimeCaps]
}

-- MAC-ALGORITHM
--
--  Describes the basic properties of a message
--      authentication code (MAC) algorithm
--
--  &id - contains the OID identifying the MAC algorithm
--  &Params - if present, contains the type for the algorithm
--               parameters; if absent, implies no parameters
--  &paramPresence - parameter presence requirement
--  &keyed - MAC algorithm is a keyed MAC algorithm
--  &smimeCaps - contains the object describing how the S/MIME
--              capabilities are presented.
```

```
--
--  Some parameters that perhaps should have been added would be
--  fields with the minimum and maximum MAC lengths for
--  those MAC algorithms that allow truncations.
--
--  Example:
--  maca-hmac-sha1 MAC-ALGORITHM ::= {
--      IDENTIFIER hMAC-SHA1
--      PARAMS TYPE NULL ARE preferredAbsent
--      IS KEYED MAC TRUE
--      SMIME-CAPS {IDENTIFIED BY hMAC-SHA1}
--  }

MAC-ALGORITHM ::= CLASS {
    &id                 OBJECT IDENTIFIER UNIQUE,
    &Params             OPTIONAL,
    &paramPresence      ParamOptions DEFAULT absent,
    &keyed              BOOLEAN,
    &smimeCaps          SMIME-CAPS OPTIONAL
} WITH SYNTAX {
    IDENTIFIER &id
    [PARAMS [TYPE &Params] ARE &paramPresence]
    IS-KEYED-MAC &keyed
    [SMIME-CAPS &smimeCaps]
}

--  CONTENT-ENCRYPTION
--
--  Describes the basic properties of a content encryption
--      algorithm
--
--  &id - contains the OID identifying the content
--        encryption algorithm
--  &Params - if present, contains the type for the algorithm
--               parameters; if absent, implies no parameters
--  &paramPresence - parameter presence requirement
--  &smimeCaps - contains the object describing how the S/MIME
--              capabilities are presented.
--
--  Example:
--  cea-3DES-cbc CONTENT-ENCRYPTION ::= {
--      IDENTIFIER des-ede3-cbc
--      PARAMS TYPE IV ARE required
--      SMIME-CAPS { IDENTIFIED BY des-ede3-cbc }
--  }

CONTENT-ENCRYPTION ::= CLASS {
    &id                OBJECT IDENTIFIER UNIQUE,
    &Params             OPTIONAL,
    &paramPresence      ParamOptions DEFAULT absent,
    &smimeCaps          SMIME-CAPS OPTIONAL
} WITH SYNTAX {
    IDENTIFIER &id
    [PARAMS [TYPE &Params] ARE &paramPresence]
    [SMIME-CAPS &smimeCaps]
}

-- ALGORITHM
--
-- Describes a generic algorithm identifier
--
--  &id - contains the OID identifying the algorithm
--  &Params - if present, contains the type for the algorithm
--               parameters; if absent, implies no parameters
--  &paramPresence - parameter presence requirement
--  &smimeCaps - contains the object describing how the S/MIME
--              capabilities are presented.
--
--  This would be used for cases where an algorithm of an unknown
--  type is used.  In general however, one should either define
--  a more complete algorithm structure (such as the one above)
--  or use the TYPE-IDENTIFIER class.

ALGORITHM ::= CLASS {
    &id OBJECT   IDENTIFIER UNIQUE,
    &Params      OPTIONAL,
    &paramPresence ParamOptions DEFAULT absent,
    &smimeCaps   SMIME-CAPS OPTIONAL
} WITH SYNTAX {
    IDENTIFIER &id
    [PARAMS [TYPE &Params] ARE &paramPresence]
    [SMIME-CAPS &smimeCaps]
}

-- AlgorithmIdentifier
--
-- Provides the generic structure that is used to encode algorithm
--    identification and the parameters associated with the
--    algorithm.
--
-- The first parameter represents the type of the algorithm being
--    used.
-- The second parameter represents an object set containing the
--    algorithms that may occur in this situation.
--    The initial list of required algorithms should occur to the
--      left of an extension marker; all other algorithms should
--      occur to the right of an extension marker.
--
-- The object class ALGORITHM can be used for generic unspecified
--     items.
-- If new ALGORITHM classes are defined, the fields &id and &Params
--     need to be present as fields in the object in order to use
--     this parameterized type.
--
-- Example:
--    SignatureAlgorithmIdentifier ::=
--       AlgorithmIdentifier{SIGNATURE-ALGORITHM, {SignatureAlgSet}}

AlgorithmIdentifier{ALGORITHM-TYPE, ALGORITHM-TYPE:AlgorithmSet} ::=
        SEQUENCE {
            algorithm   ALGORITHM-TYPE.&id({AlgorithmSet}),
            parameters  ALGORITHM-TYPE.
                   &Params({AlgorithmSet}{@algorithm}) OPTIONAL
        }

--  S/MIME Capabilities
--
--  We have moved the SMIME-CAPS from the module for RFC 3851 to here
--  because it is used in RFC 4262 (X.509 Certificate Extension for
--  S/MIME Capabilities)
--
--
--  This class is used to represent an S/MIME capability.  S/MIME
--  capabilities are used to represent what algorithm capabilities
--  an individual has.  The classic example was the content encryption
--  algorithm RC2 where the algorithm id and the RC2 key lengths
--  supported needed to be advertised, but the IV used is not fixed.
--  Thus, for RC2 we used
--
--  cap-RC2CBC SMIME-CAPS ::= {
--      TYPE INTEGER ( 40 | 128 ) IDENTIFIED BY rc2-cbc }
--
--  where 40 and 128 represent the RC2 key length in number of bits.
--
--  Another example where information needs to be shown is for
--  RSA-OAEP where only specific hash functions or mask generation
--  functions are supported, but the saltLength is specified by the
--  sender and not the recipient.  In this case, one can either
--  generate a number of capability items,
--  or a new S/MIME capability type could be generated where
--  multiple hash functions could be specified.
--
--
--  SMIME-CAP
```

```
--
--  This class is used to associate the type that describes the
--  capabilities with the object identifier.
--

SMIME-CAPS ::= CLASS {
    &id         OBJECT IDENTIFIER UNIQUE,
    &Type       OPTIONAL
}
WITH SYNTAX { [TYPE &Type] IDENTIFIED BY &id }

--
--  Generic type - this is used for defining values.
--

--  Define a single S/MIME capability encoding

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

--  Define a sequence of S/MIME capability values

SMIMECapabilities { SMIME-CAPS:CapabilitySet } ::=
        SEQUENCE SIZE (1..MAX) OF SMIMECapability{{CapabilitySet} }

END
```

3. ASN.1 Module for RFC 3370

```
   CryptographicMessageSyntaxAlgorithms-2009
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
       smime(16) modules(0) id-mod-cmsalg-2001-02(37) }
   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN
   IMPORTS

   ParamOptions, DIGEST-ALGORITHM, SIGNATURE-ALGORITHM,
      PUBLIC-KEY, KEY-DERIVATION, KEY-WRAP, MAC-ALGORITHM,
      KEY-AGREE, KEY-TRANSPORT, CONTENT-ENCRYPTION, ALGORITHM,
      AlgorithmIdentifier{}, SMIME-CAPS
   FROM AlgorithmInformation-2009
       {iso(1) identified-organization(3) dod(6) internet(1) security(5)
       mechanisms(5) pkix(7) id-mod(0)
       id-mod-algorithmInformation-02(58)}
   pk-rsa, pk-dh, pk-dsa, rsaEncryption, DHPublicKey, dhpublicnumber
   FROM PKIXAlgs-2009
        {iso(1) identified-organization(3) dod(6)
        internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-pkix1-algorithms2008-02(56)}

   cap-RC2CBC
   FROM SecureMimeMessageV3dot1-2009
        {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
        smime(16) modules(0) id-mod-msg-v3dot1-02(39)};

   --  2. Hash algorithms in this document

   MessageDigestAlgs DIGEST-ALGORITHM ::= {
   --   mda-md5 | mda-sha1,
        ... }

   --  3. Signature algorithms in this document

   SignatureAlgs SIGNATURE-ALGORITHM ::= {
   --  See RFC 3279
   --  sa-dsaWithSHA1 |  sa-rsaWithMD5 | sa-rsaWithSHA1,
       ... }

   --  4.  Key Management Algorithms
   --  4.1 Key Agreement Algorithms

   KeyAgreementAlgs KEY-AGREE ::= { kaa-esdh | kaa-ssdh, ...}
   KeyAgreePublicKeys PUBLIC-KEY ::= { pk-dh, ...}

   --  4.2  Key Transport Algorithms

   KeyTransportAlgs KEY-TRANSPORT ::= { kt-rsa, ... }

   --  4.3  Symmetric Key-Encryption Key Algorithms

   KeyWrapAlgs KEY-WRAP ::= { kwa-3DESWrap | kwa-RC2Wrap, ... }

   --  4.4  Key Derivation Algorithms

   KeyDerivationAlgs KEY-DERIVATION ::= { kda-PBKDF2, ... }

   --  5.  Content Encryption Algorithms

   ContentEncryptionAlgs CONTENT-ENCRYPTION ::=
       { cea-3DES-cbc | cea-RC2-cbc, ... }

   --  6.  Message Authentication Code Algorithms
   MessageAuthAlgs MAC-ALGORITHM ::= { maca-hMAC-SHA1, ... }

   --  S/MIME Capabilities for these items

   SMimeCaps SMIME-CAPS ::= {
       kaa-esdh.&smimeCaps          |
       kaa-ssdh.&smimeCaps          |
       kt-rsa.&smimeCaps            |
       kwa-3DESWrap.&smimeCaps      |
       kwa-RC2Wrap.&smimeCaps       |
       cea-3DES-cbc.&smimeCaps      |
       cea-RC2-cbc.&smimeCaps       |
       maca-hMAC-SHA1.&smimeCaps,
       ...}

   --
   --
   --

   -- Algorithm Identifiers

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

   KeyWrapAlgorithm ::=
       AlgorithmIdentifier {KEY-WRAP, {KeyWrapAlgs }}

   RC2wrapParameter ::= RC2ParameterVersion
   RC2ParameterVersion ::= INTEGER

   CBCParameter ::= IV

   IV ::= OCTET STRING  -- exactly 8 octets

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

   maca-hMAC-SHA1 MAC-ALGORITHM ::= {
       IDENTIFIER hMAC-SHA1
       PARAMS TYPE NULL ARE preferredAbsent
       IS-KEYED-MAC TRUE
       SMIME-CAPS {IDENTIFIED BY hMAC-SHA1}
   }

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

   alg-hMAC-SHA1 ALGORITHM ::=
       { IDENTIFIER hMAC-SHA1 PARAMS TYPE NULL ARE required }

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

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

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

   defaultPBKDF2 PBKDF2-PRFsAlgorithmIdentifier ::=
       { algorithm alg-hMAC-SHA1.&id, parameters NULL:NULL }

   PBKDF2-params ::= SEQUENCE {
       salt CHOICE {
           specified OCTET STRING,
           otherSource PBKDF2-SaltSourcesAlgorithmIdentifier },
       iterationCount INTEGER (1..MAX),
       keyLength INTEGER (1..MAX) OPTIONAL,
       prf PBKDF2-PRFsAlgorithmIdentifier DEFAULT
               defaultPBKDF2
           }
   --
   --  This object is included for completeness.  It should not be used
   --      for encoding of signatures, but was sometimes used in older
   --      versions of CMS for encoding of RSA signatures.
   --
   --
   -- sa-rsa SIGNATURE-ALGORITHM ::= {
   --         IDENTIFIER rsaEncryption
   --         - - value is not ASN.1 encoded
   --         PARAMS TYPE NULL ARE required
   --         HASHES {mda-sha1 | mda-md5, ...}
   --         PUBLIC-KEYS { pk-rsa}
   -- }
   --
   -- No ASN.1 encoding is applied to the signature value
   --    for these items

   kaa-esdh KEY-AGREE ::= {
        IDENTIFIER id-alg-ESDH
        PARAMS TYPE KeyWrapAlgorithm ARE required
        PUBLIC-KEYS { pk-dh }
        -- UKM is not ASN.1 encoded
        UKM ARE optional
        SMIME-CAPS {TYPE KeyWrapAlgorithm IDENTIFIED BY id-alg-ESDH}
   }

   kaa-ssdh KEY-AGREE ::= {
        IDENTIFIER id-alg-SSDH
        PARAMS TYPE KeyWrapAlgorithm ARE required
        PUBLIC-KEYS {pk-dh}
        -- UKM is not ASN.1 encoded
        UKM ARE optional
        SMIME-CAPS {TYPE KeyWrapAlgorithm IDENTIFIED BY id-alg-SSDH}
   }

   dh-public-number OBJECT IDENTIFIER ::= dhpublicnumber

   pk-originator-dh PUBLIC-KEY ::= {
       IDENTIFIER dh-public-number
       KEY DHPublicKey
       PARAMS ARE absent
       CERT-KEY-USAGE {keyAgreement, encipherOnly, decipherOnly}
   }

   kwa-3DESWrap KEY-WRAP ::= {
        IDENTIFIER id-alg-CMS3DESwrap
        PARAMS TYPE NULL ARE required
        SMIME-CAPS {IDENTIFIED BY id-alg-CMS3DESwrap}
   }

   kwa-RC2Wrap KEY-WRAP ::= {
        IDENTIFIER id-alg-CMSRC2wrap
        PARAMS TYPE RC2wrapParameter ARE required
        SMIME-CAPS { IDENTIFIED BY id-alg-CMSRC2wrap }
   }

   kda-PBKDF2 KEY-DERIVATION ::= {
       IDENTIFIER id-PBKDF2
       PARAMS TYPE PBKDF2-params ARE required
       -- No S/MIME caps defined
   }

   cea-3DES-cbc CONTENT-ENCRYPTION ::= {
       IDENTIFIER des-ede3-cbc
       PARAMS TYPE IV ARE required
       SMIME-CAPS { IDENTIFIED BY des-ede3-cbc }
   }

   cea-RC2-cbc CONTENT-ENCRYPTION ::= {
       IDENTIFIER rc2-cbc
       PARAMS TYPE RC2CBCParameter ARE required
       SMIME-CAPS cap-RC2CBC
   }

   kt-rsa KEY-TRANSPORT ::= {
       IDENTIFIER rsaEncryption
       PARAMS TYPE NULL ARE required
       PUBLIC-KEYS { pk-rsa }
       SMIME-CAPS {IDENTIFIED BY rsaEncryption}
   }

   --  S/MIME Capabilities - most have no label.

   cap-3DESwrap SMIME-CAPS ::= { IDENTIFIED BY id-alg-CMS3DESwrap }

   END
```

4. ASN.1 Module for RFC 3565

```
   CMSAesRsaesOaep-2009 {iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-aes-02(38)}
   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN
   IMPORTS

   CONTENT-ENCRYPTION, KEY-WRAP, SMIME-CAPS
   FROM AlgorithmInformation-2009
       {iso(1) identified-organization(3) dod(6) internet(1) security(5)
       mechanisms(5) pkix(7) id-mod(0)
       id-mod-algorithmInformation-02(58)};

   AES-ContentEncryption CONTENT-ENCRYPTION ::= {
       cea-aes128-cbc | cea-aes192-cbc | cea-aes256-cbc, ...
   }

   AES-KeyWrap KEY-WRAP ::= {
       kwa-aes128-wrap | kwa-aes192-wrap | kwa-aes256-wrap, ...
   }

   SMimeCaps SMIME-CAPS ::= {
      cea-aes128-cbc.&smimeCaps |
      cea-aes192-cbc.&smimeCaps |
      cea-aes256-cbc.&smimeCaps |
      kwa-aes128-wrap.&smimeCaps |
      kwa-aes192-wrap.&smimeCaps |
      kwa-aes256-wrap.&smimeCaps, ...
   }

   -- AES information object identifiers --

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

   -- AES using CBC mode for key sizes of 128, 192, 256

   cea-aes128-cbc CONTENT-ENCRYPTION ::= {
       IDENTIFIER id-aes128-CBC
       PARAMS TYPE AES-IV ARE required
       SMIME-CAPS { IDENTIFIED BY id-aes128-CBC }
   }
   id-aes128-CBC OBJECT IDENTIFIER ::= { aes 2 }

   cea-aes192-cbc CONTENT-ENCRYPTION ::= {
       IDENTIFIER id-aes192-CBC
```

```
        PARAMS TYPE AES-IV ARE required
        SMIME-CAPS { IDENTIFIED BY id-aes192-CBC }
    }
    id-aes192-CBC OBJECT IDENTIFIER ::= { aes 22 }

    cea-aes256-cbc CONTENT-ENCRYPTION ::= {
        IDENTIFIER id-aes256-CBC
        PARAMS TYPE AES-IV ARE required
        SMIME-CAPS { IDENTIFIED BY id-aes256-CBC }
    }
    id-aes256-CBC OBJECT IDENTIFIER ::= { aes 42 }

    -- AES-IV is the parameter for all the above object identifiers.

    AES-IV ::= OCTET STRING (SIZE(16))

    -- AES Key Wrap Algorithm Identifiers  - Parameter is absent

    kwa-aes128-wrap KEY-WRAP ::= {
        IDENTIFIER id-aes128-wrap
        PARAMS ARE absent
        SMIME-CAPS { IDENTIFIED BY id-aes128-wrap }
    }
    id-aes128-wrap OBJECT IDENTIFIER ::= { aes 5 }

    kwa-aes192-wrap KEY-WRAP ::= {
        IDENTIFIER id-aes192-wrap
        PARAMS ARE absent
        SMIME-CAPS { IDENTIFIED BY id-aes192-wrap }
    }
    id-aes192-wrap OBJECT IDENTIFIER ::= { aes 25 }

    kwa-aes256-wrap KEY-WRAP ::= {
        IDENTIFIER id-aes256-wrap
        PARAMS ARE absent
        SMIME-CAPS { IDENTIFIED BY id-aes256-wrap }
    }
    id-aes256-wrap OBJECT IDENTIFIER ::= { aes 45 }

    END
```

5. ASN.1 Module for RFC 3851

```
 SecureMimeMessageV3dot1-2009
       {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
       smime(16) modules(0) id-mod-msg-v3dot1-02(39)}
 DEFINITIONS IMPLICIT TAGS ::=
 BEGIN
 IMPORTS

 SMIME-CAPS, SMIMECapabilities{}
 FROM AlgorithmInformation-2009
     {iso(1) identified-organization(3) dod(6) internet(1) security(5)
     mechanisms(5) pkix(7) id-mod(0)
     id-mod-algorithmInformation-02(58)}

 ATTRIBUTE
 FROM PKIX-CommonTypes-2009
     {iso(1) identified-organization(3) dod(6) internet(1) security(5)
     mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57)}

 SubjectKeyIdentifier, IssuerAndSerialNumber, RecipientKeyIdentifier
 FROM CryptographicMessageSyntax-2009
     {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
     smime(16) modules(0) id-mod-cms-2004-02(41)}

 rc2-cbc, SMimeCaps
 FROM CryptographicMessageSyntaxAlgorithms-2009
     {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
     smime(16) modules(0) id-mod-cmsalg-2001-02(37)}

 SMimeCaps
 FROM PKIXAlgs-2009
     {iso(1) identified-organization(3) dod(6) internet(1) security(5)
     mechanisms(5) pkix(7) id-mod(0)
     id-mod-pkix1-algorithms2008-02(56)}

 SMimeCaps
 FROM PKIX1-PSS-OAEP-Algorithms-2009
      {iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-rsa-pkalgs-02(54)};

 SMimeAttributeSet ATTRIBUTE ::=
     { aa-smimeCapabilities | aa-encrypKeyPref, ... }

 --  id-aa is the arc with all new authenticated and unauthenticated
 --  attributes produced by the 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)}

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

 aa-smimeCapabilities ATTRIBUTE ::=
     { TYPE SMIMECapabilities{{SMimeCapsSet}} IDENTIFIED BY
           smimeCapabilities }
 smimeCapabilities OBJECT IDENTIFIER ::=
     { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
     15 }

 SMimeCapsSet SMIME-CAPS ::=
     { cap-preferBinaryInside | cap-RC2CBC |
     PKIXAlgs-2009.SMimeCaps |
     CryptographicMessageSyntaxAlgorithms-2009.SMimeCaps |
     PKIX1-PSS-OAEP-Algorithms-2009.SMimeCaps, ... }

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

 aa-encrypKeyPref ATTRIBUTE ::=
     { TYPE SMIMEEncryptionKeyPreference
         IDENTIFIED BY id-aa-encrypKeyPref }

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

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

 -- receipentKeyId is spelt incorrectly, but kept for historical
 -- reasons.

 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

 cap-preferBinaryInside SMIME-CAPS ::=
     { -- No value -- IDENTIFIED BY id-cap-preferBinaryInside }

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

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

 -- Signature Algorithms Not Found in [RFC3370]
 --
 -- 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 [RFC5652] for a description of how to encode the attribute
 --    value.

 cap-RC2CBC SMIME-CAPS ::=
     { TYPE SMIMECapabilitiesParametersForRC2CBC
         IDENTIFIED BY rc2-cbc}

 SMIMECapabilitiesParametersForRC2CBC ::= INTEGER (40 | 128, ...)
 --    (RC2 Key Length (number of bits))

 END
```

6. ASN.1 Module for RFC 3852

This module has an ASN.1 idiom for noting in which version of CMS changes were made from the original PKCS #7; that idiom is "[[v:", where "v" is an integer. For example:

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

Similarly, this module adds the ASN.1 idiom for extensibility (the "...,") in all places that have been extended in the past. See the example above.

```
CryptographicMessageSyntax-2009
    { iso(1) member-body(2) us(840) rsadsi(113549)
    pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2004-02(41) }
DEFINITIONS IMPLICIT TAGS ::=
BEGIN
IMPORTS

ParamOptions, DIGEST-ALGORITHM, SIGNATURE-ALGORITHM,
    PUBLIC-KEY, KEY-DERIVATION, KEY-WRAP, MAC-ALGORITHM,
    KEY-AGREE, KEY-TRANSPORT, CONTENT-ENCRYPTION, ALGORITHM,
    AlgorithmIdentifier
FROM AlgorithmInformation-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0)
    id-mod-algorithmInformation-02(58)}
SignatureAlgs, MessageDigestAlgs, KeyAgreementAlgs,
    MessageAuthAlgs, KeyWrapAlgs, ContentEncryptionAlgs,
    KeyTransportAlgs, KeyDerivationAlgs, KeyAgreePublicKeys
FROM CryptographicMessageSyntaxAlgorithms-2009
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) modules(0) id-mod-cmsalg-2001-02(37) }

Certificate, CertificateList, CertificateSerialNumber,
    Name, ATTRIBUTE
FROM PKIX1Explicit-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-pkix1-explicit-02(51) }

AttributeCertificate
FROM PKIXAttributeCertificate-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0)
    id-mod-attribute-cert-02(47) }

AttributeCertificateV1
FROM AttributeCertificateVersion1-2009
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) modules(0) id-mod-v1AttrCert-02(49) } ;

-- Cryptographic Message Syntax

-- The following are used for version numbers using the ASN.1
--   idiom "[[n:"
--   Version 1 = PKCS #7
--   Version 2 = S/MIME V2
--   Version 3 = RFC 2630
--   Version 4 = RFC 3369
--   Version 5 = RFC 3852

CONTENT-TYPE ::= TYPE-IDENTIFIER
ContentType ::= CONTENT-TYPE.&id
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 }

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 SignedAttributes OPTIONAL,
    signatureAlgorithm SignatureAlgorithmIdentifier,
    signature SignatureValue,
    unsignedAttrs [1] IMPLICIT Attributes
        {{UnsignedAttributes}} OPTIONAL }

SignedAttributes ::= Attributes {{ SignedAttributesSet }}

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

SignedAttributesSet ATTRIBUTE ::=
    { aa-signingTime | aa-messageDigest | aa-contentType, ... }

UnsignedAttributes ATTRIBUTE ::= { aa-countersignature, ... }
SignatureValue ::= OCTET STRING

EnvelopedData ::= SEQUENCE {
    version CMSVersion,
    originatorInfo [0] IMPLICIT OriginatorInfo OPTIONAL,
    recipientInfos RecipientInfos,
    encryptedContentInfo EncryptedContentInfo,
    ...,
    [[2: 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 } )

UnprotectedAttributes ATTRIBUTE ::=  { ... }

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

EncryptedKey ::= OCTET STRING

KeyTransRecipientInfo ::= SEQUENCE {
    version CMSVersion,  -- always set to 0 or 2
    rid RecipientIdentifier,
    keyEncryptionAlgorithm AlgorithmIdentifier
        {KEY-TRANSPORT, {KeyTransportAlgorithmSet}},
    encryptedKey EncryptedKey }

KeyTransportAlgorithmSet KEY-TRANSPORT ::= { KeyTransportAlgs, ... }

RecipientIdentifier ::= CHOICE {
    issuerAndSerialNumber IssuerAndSerialNumber,
    ...,
    [[2: subjectKeyIdentifier [0] SubjectKeyIdentifier ]] }
KeyAgreeRecipientInfo ::= SEQUENCE {
    version CMSVersion,  -- always set to 3
    originator [0] EXPLICIT OriginatorIdentifierOrKey,
    ukm [1] EXPLICIT UserKeyingMaterial OPTIONAL,
    keyEncryptionAlgorithm AlgorithmIdentifier
        {KEY-AGREE, {KeyAgreementAlgorithmSet}},
    recipientEncryptedKeys RecipientEncryptedKeys }

KeyAgreementAlgorithmSet KEY-AGREE ::= { KeyAgreementAlgs, ... }

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

OriginatorPublicKey ::= SEQUENCE {
    algorithm AlgorithmIdentifier {PUBLIC-KEY, {OriginatorKeySet}},
    publicKey BIT STRING }

OriginatorKeySet PUBLIC-KEY ::= { KeyAgreePublicKeys, ... }

RecipientEncryptedKeys ::= SEQUENCE OF RecipientEncryptedKey

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

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,
    ...,
    [[2: 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
    {{AuthAttributeSet}}

AuthAttributeSet ATTRIBUTE ::= { aa-contentType | aa-messageDigest
                                     | aa-signingTime, ...}
MessageAuthenticationCode ::= OCTET STRING

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

UnauthAttributeSet ATTRIBUTE ::= {...}

--
--  General algorithm definitions
--

DigestAlgorithmIdentifier ::= AlgorithmIdentifier
    {DIGEST-ALGORITHM, {DigestAlgorithmSet}}

DigestAlgorithmSet DIGEST-ALGORITHM ::= {
    CryptographicMessageSyntaxAlgorithms-2009.MessageDigestAlgs, ... }

SignatureAlgorithmIdentifier ::= AlgorithmIdentifier
    {SIGNATURE-ALGORITHM, {SignatureAlgorithmSet}}

SignatureAlgorithmSet SIGNATURE-ALGORITHM ::=
    { SignatureAlgs, ... }

KeyEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
    {KEY-WRAP, {KeyEncryptionAlgorithmSet}}

KeyEncryptionAlgorithmSet KEY-WRAP ::= { KeyWrapAlgs, ... }

ContentEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
    {CONTENT-ENCRYPTION, {ContentEncryptionAlgorithmSet}}

ContentEncryptionAlgorithmSet CONTENT-ENCRYPTION ::=
    { ContentEncryptionAlgs, ... }

MessageAuthenticationCodeAlgorithm ::= AlgorithmIdentifier
    {MAC-ALGORITHM, {MessageAuthenticationCodeAlgorithmSet}}

MessageAuthenticationCodeAlgorithmSet MAC-ALGORITHM ::=
    { MessageAuthAlgs, ... }
KeyDerivationAlgorithmIdentifier ::= AlgorithmIdentifier
    {KEY-DERIVATION, {KeyDerivationAlgs, ...}}

RevocationInfoChoices ::= SET OF RevocationInfoChoice

RevocationInfoChoice ::= CHOICE {
    crl CertificateList,
    ...,
    [[5: 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
    ...,
    [[3: v1AttrCert [1] IMPLICIT AttributeCertificateV1]],
         -- Obsolete
    [[4: v2AttrCert [2] IMPLICIT AttributeCertificateV2]],
    [[5: 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

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

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

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

aa-countersignature ATTRIBUTE ::=
    { TYPE Countersignature 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

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

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

END
```

7. ASN.1 Module for RFC 4108

```
  CMSFirmwareWrapper-2009
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) id-mod-cms-firmware-wrap-02(40) }
  DEFINITIONS IMPLICIT TAGS ::=
  BEGIN
  IMPORTS

  OTHER-NAME
  FROM PKIX1Implicit-2009
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59) }

  EnvelopedData, CONTENT-TYPE, ATTRIBUTE
  FROM CryptographicMessageSyntax-2009
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) id-mod-cms-2004-02(41) };

  FirmwareContentTypes CONTENT-TYPE ::= {
          ct-firmwarePackage | ct-firmwareLoadReceipt |
          ct-firmwareLoadError,... }
  FirmwareSignedAttrs ATTRIBUTE ::= {
          aa-firmwarePackageID | aa-targetHardwareIDs |
          aa-decryptKeyID | aa-implCryptoAlgs | aa-implCompressAlgs |
          aa-communityIdentifiers | aa-firmwarePackageInfo,... }
  FirmwareUnsignedAttrs ATTRIBUTE ::= {
          aa-wrappedFirmwareKey, ... }

  FirmwareOtherNames OTHER-NAME ::= {
          on-hardwareModuleName, ... }

  -- Firmware Package Content Type and Object Identifier

  ct-firmwarePackage CONTENT-TYPE ::=
          { FirmwarePkgData IDENTIFIED BY id-ct-firmwarePackage }

  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

  aa-firmwarePackageID ATTRIBUTE ::=
      { TYPE FirmwarePackageIdentifier IDENTIFIED BY
          id-aa-firmwarePackageID }

  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 }

  aa-targetHardwareIDs ATTRIBUTE ::=
      { TYPE TargetHardwareIdentifiers IDENTIFIED BY
          id-aa-targetHardwareIDs }

  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

  aa-decryptKeyID ATTRIBUTE ::=
          { TYPE DecryptKeyIdentifier IDENTIFIED BY id-aa-decryptKeyID}

  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

  aa-implCryptoAlgs ATTRIBUTE ::=
      { TYPE ImplementedCryptoAlgorithms IDENTIFIED BY
          id-aa-implCryptoAlgs }

  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

  aa-implCompressAlgs ATTRIBUTE ::=
      { TYPE ImplementedCompressAlgorithms IDENTIFIED BY
          id-aa-implCompressAlgs }

  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

  aa-communityIdentifiers ATTRIBUTE ::=
      { TYPE CommunityIdentifiers IDENTIFIED BY
          id-aa-communityIdentifiers }

  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
      }
  }

  aa-firmwarePackageInfo ATTRIBUTE ::=
      { TYPE FirmwarePackageInfo IDENTIFIED BY
          id-aa-firmwarePackageInfo }
  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

  aa-wrappedFirmwareKey ATTRIBUTE ::=
      { TYPE WrappedFirmwareKey IDENTIFIED BY
          id-aa-wrappedFirmwareKey }
  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

  ct-firmwareLoadReceipt CONTENT-TYPE ::=
      { FirmwarePackageLoadReceipt IDENTIFIED BY
          id-ct-firmwareLoadReceipt }
  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

  ct-firmwareLoadError CONTENT-TYPE ::=
      { FirmwarePackageLoadError
          IDENTIFIED BY id-ct-firmwareLoadError }
  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

  on-hardwareModuleName OTHER-NAME ::=
          { HardwareModuleName IDENTIFIED BY id-on-hardwareModuleName }
  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
```

8. ASN.1 Module for RFC 4998

```
   ERS {iso(1) identified-organization(3) dod(6) internet(1)
       security(5) mechanisms(5) ltans(11) id-mod(0) id-mod-ers(1)
       id-mod-ers-v1(1) }
   DEFINITIONS IMPLICIT TAGS ::=
   BEGIN
   IMPORTS

   AttributeSet{}, ATTRIBUTE
   FROM PKIX-CommonTypes
       {iso(1) identified-organization(3) dod(6) internet(1) security(5)
       mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57) }

   AlgorithmIdentifier{}, ALGORITHM, DIGEST-ALGORITHM
   FROM AlgorithmInformation-2009
       {iso(1) identified-organization(3) dod(6) internet(1) security(5)
       mechanisms(5) pkix(7) id-mod(0)
       id-mod-algorithmInformation-02(58)}

   ContentInfo
   FROM CryptographicMessageSyntax2004
       { iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2004-02(41) } ;

   aa-er-Internal ATTRIBUTE ::=
       { TYPE EvidenceRecord IDENTIFIED BY id-aa-er-internal }
   id-aa-er-internal OBJECT IDENTIFIER ::=
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
       smime(16) id-aa(2) 49 }

   aa-er-External ATTRIBUTE ::=
       { TYPE EvidenceRecord IDENTIFIED BY id-aa-er-external }
   id-aa-er-external OBJECT IDENTIFIER ::=
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
       smime(16) id-aa(2) 50 }

   ltans OBJECT IDENTIFIER ::=
       {iso(1) identified-organization(3) dod(6) internet(1) security(5)
       mechanisms(5) ltans(11) }

   EvidenceRecord ::= SEQUENCE {
       version                   INTEGER { v1(1) } ,
       digestAlgorithms          SEQUENCE OF AlgorithmIdentifier
                                     {DIGEST-ALGORITHM, {...}},
       cryptoInfos               [0] CryptoInfos OPTIONAL,
       encryptionInfo            [1] EncryptionInfo OPTIONAL,
       archiveTimeStampSequence  ArchiveTimeStampSequence
   }

   CryptoInfos ::= SEQUENCE SIZE (1..MAX) OF AttributeSet{{...}}

   ArchiveTimeStampSequence ::= SEQUENCE OF ArchiveTimeStampChain
   ArchiveTimeStampChain    ::= SEQUENCE OF ArchiveTimeStamp

   ArchiveTimeStamp ::= SEQUENCE {
      digestAlgorithm [0] AlgorithmIdentifier{DIGEST-ALGORITHM, {...}}
                                OPTIONAL,
      attributes      [1] Attributes OPTIONAL,
      reducedHashtree [2] SEQUENCE OF PartialHashtree OPTIONAL,
      timeStamp       ContentInfo
   }

   PartialHashtree ::= SEQUENCE OF OCTET STRING

   Attributes ::= SET SIZE (1..MAX) OF AttributeSet{{...}}

   EncryptionInfo       ::=     SEQUENCE {
       encryptionInfoType   ENCINFO-TYPE.
                                &id({SupportedEncryptionAlgorithms}),
       encryptionInfoValue  ENCINFO-TYPE.
                                &Type({SupportedEncryptionAlgorithms}
                                    {@encryptionInfoType})
   }

   ENCINFO-TYPE ::= TYPE-IDENTIFIER

   SupportedEncryptionAlgorithms ENCINFO-TYPE ::= {...}

   END
```

9. ASN.1 Module for RFC 5035

Section numbers in the module refer to the sections of RFC 2634 as updated by RFC 5035.

```
  ExtendedSecurityServices-2009
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) id-mod-ess-2006-02(42) }
  DEFINITIONS IMPLICIT TAGS ::=
  BEGIN
  IMPORTS

  AttributeSet{}, ATTRIBUTE, SECURITY-CATEGORY, SecurityCategory{}
  FROM PKIX-CommonTypes-2009
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
```

```
        mechanisms(5) pkix(7) id-mod(0) id-mod-pkixCommon-02(57) }

    AlgorithmIdentifier{}, ALGORITHM, DIGEST-ALGORITHM
    FROM AlgorithmInformation-2009
        {iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) pkix(7) id-mod(0)
        id-mod-algorithmInformation-02(58)}

    ContentType, IssuerAndSerialNumber, SubjectKeyIdentifier,
        CONTENT-TYPE
    FROM CryptographicMessageSyntax-2009
        { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
        smime(16) modules(0) id-mod-cms-2004-02(41) }

    CertificateSerialNumber
    FROM PKIX1Explicit-2009
        { iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-explicit-02(51) }

    PolicyInformation, GeneralNames
    FROM PKIX1Implicit-2009
        { iso(1) identified-organization(3) dod(6) internet(1) security(5)
        mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59)}

    mda-sha256
    FROM PKIX1-PSS-OAEP-Algorithms-2009
         { iso(1) identified-organization(3) dod(6)
           internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
           id-mod-pkix1-rsa-pkalgs-02(54) } ;

    EssSignedAttributes ATTRIBUTE ::= {
        aa-receiptRequest | aa-contentIdentifier | aa-contentHint |
        aa-msgSigDigest | aa-contentReference | aa-securityLabel |
        aa-equivalentLabels | aa-mlExpandHistory | aa-signingCertificate |
        aa-signingCertificateV2, ... }

    EssContentTypes CONTENT-TYPE ::= { ct-receipt, ... }

    -- 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.

    -- Section 2.7
```

```
  aa-receiptRequest ATTRIBUTE ::=
      { TYPE ReceiptRequest IDENTIFIED BY id-aa-receiptRequest}

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

  ub-receiptsTo INTEGER ::= 16

  aa-contentIdentifier ATTRIBUTE ::=
      { TYPE ContentIdentifier IDENTIFIED BY id-aa-contentIdentifier}
  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}

  ct-receipt CONTENT-TYPE ::=
      { Receipt IDENTIFIED BY id-ct-receipt }
  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}

  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
  }

  ESSVersion ::= INTEGER  { v1(1) }
  -- Section 2.9

  aa-contentHint ATTRIBUTE ::=
      { TYPE ContentHints IDENTIFIED BY id-aa-contentHint }
  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}

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

  -- Section 2.10

  aa-msgSigDigest ATTRIBUTE ::=
      { TYPE MsgSigDigest IDENTIFIED BY id-aa-msgSigDigest }
  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}

  MsgSigDigest ::= OCTET STRING

  -- Section 2.11

  aa-contentReference ATTRIBUTE ::=
      { TYPE ContentReference IDENTIFIED BY id-aa-contentReference }
  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 }

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

  -- Section 3.2

  aa-securityLabel ATTRIBUTE ::=
      { TYPE ESSSecurityLabel IDENTIFIED BY id-aa-securityLabel }
  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}

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

```
   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
           {{SupportedSecurityCategories}}

   ub-security-categories INTEGER ::= 64

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

   -- Section 3.4

   aa-equivalentLabels ATTRIBUTE ::=
       { TYPE EquivalentLabels IDENTIFIED BY id-aa-equivalentLabels }
   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}

   EquivalentLabels ::= SEQUENCE OF ESSSecurityLabel

   -- Section 4.4

   aa-mlExpandHistory ATTRIBUTE ::=
       { TYPE MLExpansionHistory IDENTIFIED BY id-aa-mlExpandHistory }
   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 }

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

```
  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

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

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

  aa-signingCertificateV2 ATTRIBUTE ::=
      { TYPE SigningCertificateV2 IDENTIFIED BY
          id-aa-signingCertificateV2 }
  id-aa-signingCertificateV2 OBJECT IDENTIFIER ::=
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
      smime(16) id-aa(2) 47 }

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

  HashAlgorithm ::= AlgorithmIdentifier{DIGEST-ALGORITHM,
                        {mda-sha256, ...}}

  ESSCertIDv2 ::= SEQUENCE {
      hashAlgorithm    HashAlgorithm
                         DEFAULT { algorithm mda-sha256.&id },
      certHash        Hash,
      issuerSerial    IssuerSerial OPTIONAL
  }
  ESSCertID ::=  SEQUENCE {
      certHash        Hash,
      issuerSerial    IssuerSerial OPTIONAL
  }

  Hash ::= OCTET STRING

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

  END
```

10. ASN.1 Module for RFC 5083

```
    CMS-AuthEnvelopedData-2009
        {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
        smime(16) modules(0) id-mod-cms-authEnvelopedData-02(43)}
    DEFINITIONS IMPLICIT TAGS ::=
    BEGIN
    IMPORTS

    AuthAttributes, CMSVersion, EncryptedContentInfo,
        MessageAuthenticationCode, OriginatorInfo, RecipientInfos,
        UnauthAttributes, CONTENT-TYPE
    FROM CryptographicMessageSyntax-2009
        {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
        smime(16) modules(0) id-mod-cms-2004-02(41)} ;

    ContentTypes CONTENT-TYPE ::= {ct-authEnvelopedData, ... }

    ct-authEnvelopedData CONTENT-TYPE ::= {
       AuthEnvelopedData IDENTIFIED BY id-ct-authEnvelopedData
    }

    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
```

11. ASN.1 Module for RFC 5084

```
  CMS-AES-CCM-and-AES-GCM-2009
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
      pkcs-9(9) smime(16) modules(0) id-mod-cms-aes-ccm-gcm-02(44) }
  DEFINITIONS IMPLICIT TAGS ::=
  BEGIN
  EXPORTS ALL;
  IMPORTS

  CONTENT-ENCRYPTION, SMIME-CAPS
  FROM AlgorithmInformation-2009
      {iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0)
      id-mod-algorithmInformation-02(58)};

  --  Add this algorithm set to include all of the algorithms defined in
  --      this document

  ContentEncryptionAlgs CONTENT-ENCRYPTION ::= {
     cea-aes128-CCM | cea-aes192-CCM | cea-aes256-CCM |
     cea-aes128-GCM | cea-aes192-GCM | cea-aes256-GCM, ... }

  SMimeCaps SMIME-CAPS ::= {
     cea-aes128-CCM.&smimeCaps |
     cea-aes192-CCM.&smimeCaps |
     cea-aes256-CCM.&smimeCaps |
     cea-aes128-GCM.&smimeCaps |
     cea-aes192-GCM.&smimeCaps |
     cea-aes256-GCM.&smimeCaps,
     ...
  }

  --  Defining objects

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

  cea-aes128-CCM CONTENT-ENCRYPTION ::= {
          IDENTIFIER id-aes128-CCM
          PARAMS TYPE CCMParameters ARE required
```

```
           SMIME-CAPS { IDENTIFIED BY id-aes128-CCM }
   }
   id-aes128-CCM OBJECT IDENTIFIER ::= { aes 7 }

   cea-aes192-CCM CONTENT-ENCRYPTION ::= {
           IDENTIFIER id-aes192-CCM
           PARAMS TYPE CCMParameters ARE required
           SMIME-CAPS { IDENTIFIED BY id-aes192-CCM }
   }
   id-aes192-CCM OBJECT IDENTIFIER ::= { aes 27 }

   cea-aes256-CCM CONTENT-ENCRYPTION ::= {
           IDENTIFIER id-aes256-CCM
           PARAMS TYPE CCMParameters ARE required
           SMIME-CAPS { IDENTIFIED BY id-aes256-CCM }
   }
   id-aes256-CCM OBJECT IDENTIFIER ::= { aes 47 }

   cea-aes128-GCM CONTENT-ENCRYPTION ::= {
           IDENTIFIER id-aes128-GCM
           PARAMS TYPE GCMParameters ARE required
           SMIME-CAPS { IDENTIFIED BY id-aes128-GCM }
   }
   id-aes128-GCM OBJECT IDENTIFIER ::= { aes 6 }

   cea-aes192-GCM CONTENT-ENCRYPTION ::= {
           IDENTIFIER id-aes128-GCM
           PARAMS TYPE GCMParameters ARE required
           SMIME-CAPS { IDENTIFIED BY id-aes192-GCM }
   }
   id-aes192-GCM OBJECT IDENTIFIER ::= { aes 26 }

   cea-aes256-GCM CONTENT-ENCRYPTION ::= {
           IDENTIFIER id-aes128-GCM
           PARAMS TYPE GCMParameters ARE required
           SMIME-CAPS { IDENTIFIED BY id-aes256-GCM }
   }
   id-aes256-GCM OBJECT IDENTIFIER ::= { aes 46 }

   -- Parameters for AlgorithmIdentifier

   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
```

12. ASN.1 Module for RFC 5275

```
  SMIMESymmetricKeyDistribution-2009
      {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) id-mod-symkeydist-02(36)}
  DEFINITIONS IMPLICIT TAGS ::=
  BEGIN
  EXPORTS ALL;
  IMPORTS

  AlgorithmIdentifier{}, ALGORITHM, DIGEST-ALGORITHM, KEY-WRAP,
          SMIMECapability{}, SMIMECapabilities{}, SMIME-CAPS
  FROM AlgorithmInformation-2009
      {iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0)
      id-mod-algorithmInformation-02(58)}

  GeneralName
  FROM PKIX1Implicit-2009
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-implicit-02(59) }

  Certificate
  FROM PKIX1Explicit-2009
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-explicit-02(51) }

  RecipientInfos, KEKIdentifier,CertificateSet
  FROM CryptographicMessageSyntax-2009
      {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) id-mod-cms-2004-02(41) }

  cap-3DESwrap
  FROM CryptographicMessageSyntaxAlgorithms
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) id-mod-cmsalg-2001-02(37) }

  AttributeCertificate
  FROM PKIXAttributeCertificate-2009
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0) id-mod-attribute-cert-02(47) }
  CMC-CONTROL, EXTENDED-FAILURE-INFO
  FROM EnrollmentMessageSyntax
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) id-mod(0) id-mod-cmc2002-02(53) }

  kwa-aes128-wrap, kwa-aes192-wrap, kwa-aes256-wrap
  FROM CMSAesRsaesOaep-2009
      { iso(1) member-body(2) us(840) rsadsi(113549)
      pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-aes-02(38) } ;

  -- This defines the group list (GL symmetric key distribution OID arc
  id-skd OBJECT IDENTIFIER ::=
      { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) skd(8) }

  SKD-ControlSet CMC-CONTROL ::= {
      skd-glUseKEK | skd-glDelete | skd-glAddMember |
      skd-glDeleteMember | skd-glRekey | skd-glAddOwner |
      skd-glRemoveOwner | skd-glKeyCompromise |
      skd-glkRefresh | skd-glaQueryRequest | skd-glProvideCert |
      skd-glManageCert | skd-glKey, ... }

  -- This defines the GL Use KEK control attribute

  skd-glUseKEK CMC-CONTROL ::=
      { GLUseKEK IDENTIFIED BY id-skd-glUseKEK }

  id-skd-glUseKEK OBJECT IDENTIFIER ::= { id-skd 1}

  GLUseKEK ::= SEQUENCE {
      glInfo            GLInfo,
      glOwnerInfo       SEQUENCE SIZE (1..MAX) OF GLOwnerInfo,
      glAdministration  GLAdministration DEFAULT managed,
      glKeyAttributes   GLKeyAttributes OPTIONAL
  }

  GLInfo ::= SEQUENCE {
      glName     GeneralName,
      glAddress  GeneralName
  }

  GLOwnerInfo ::= SEQUENCE {
      glOwnerName     GeneralName,
      glOwnerAddress  GeneralName,
      certificates    Certificates OPTIONAL
  }

  GLAdministration ::= INTEGER {
      unmanaged  (0),
      managed    (1),
      closed     (2)
  }

  --
  --  The advertised set of algorithm capabilities for the document
  --

  SKD-Caps SMIME-CAPS ::= {
      cap-3DESwrap | kwa-aes128-wrap.&smimeCaps |
      kwa-aes192-wrap.&smimeCaps | kwa-aes256-wrap.&smimeCaps,  ...
  }

  cap-aes128-cbc KeyWrapAlgorithm ::=
      { capabilityID kwa-aes128-wrap.&smimeCaps.&id }

  --
  --  The set of key wrap algorithms supported by this specification
  --

  KeyWrapAlgorithm ::= SMIMECapability{{SKD-Caps}}

  GLKeyAttributes ::= SEQUENCE {
      rekeyControlledByGLO       [0] BOOLEAN DEFAULT FALSE,
      recipientsNotMutuallyAware [1] BOOLEAN DEFAULT TRUE,
      duration                   [2] INTEGER DEFAULT 0,
      generationCounter          [3] INTEGER DEFAULT 2,
      requestedAlgorithm         [4] KeyWrapAlgorithm
                           DEFAULT cap-aes128-cbc
  }

  -- This defines the Delete GL control attribute.
  -- It has the simple type GeneralName.

  skd-glDelete CMC-CONTROL ::=
      { DeleteGL IDENTIFIED BY id-skd-glDelete }

  id-skd-glDelete OBJECT IDENTIFIER ::= { id-skd 2}
  DeleteGL ::= GeneralName

  -- This defines the Add GL Member control attribute

  skd-glAddMember CMC-CONTROL ::=
      { GLAddMember IDENTIFIED BY id-skd-glAddMember }

  id-skd-glAddMember OBJECT IDENTIFIER ::= { id-skd 3}
  GLAddMember ::= SEQUENCE {
      glName    GeneralName,
      glMember  GLMember
  }

  GLMember ::= SEQUENCE {
      glMemberName     GeneralName,
      glMemberAddress  GeneralName OPTIONAL,
      certificates     Certificates OPTIONAL
  }

  Certificates ::= SEQUENCE {
      pKC       [0] Certificate OPTIONAL,
                    -- See RFC 5280
      aC        [1] SEQUENCE SIZE (1.. MAX) OF
                      AttributeCertificate OPTIONAL,
                    -- See RFC 3281
      certPath  [2] CertificateSet OPTIONAL
                    -- From RFC 3852
  }

  -- This defines the Delete GL Member control attribute

  skd-glDeleteMember CMC-CONTROL ::=
      { GLDeleteMember IDENTIFIED BY id-skd-glDeleteMember }

  id-skd-glDeleteMember OBJECT IDENTIFIER ::= { id-skd 4}

  GLDeleteMember ::= SEQUENCE {
      glName            GeneralName,
      glMemberToDelete  GeneralName
  }

  -- This defines the Delete GL Member control attribute

  skd-glRekey CMC-CONTROL ::=
      { GLRekey IDENTIFIED BY id-skd-glRekey }

  id-skd-glRekey OBJECT IDENTIFIER ::= { id-skd 5}

  GLRekey ::= SEQUENCE {
      glName              GeneralName,
      glAdministration    GLAdministration OPTIONAL,
      glNewKeyAttributes  GLNewKeyAttributes OPTIONAL,
      glRekeyAllGLKeys    BOOLEAN OPTIONAL
  }

  GLNewKeyAttributes ::= SEQUENCE {
      rekeyControlledByGLO       [0] BOOLEAN OPTIONAL,
      recipientsNotMutuallyAware [1] BOOLEAN OPTIONAL,
      duration                   [2] INTEGER OPTIONAL,
      generationCounter          [3] INTEGER OPTIONAL,
      requestedAlgorithm         [4] KeyWrapAlgorithm OPTIONAL
  }

  -- This defines the Add and Delete GL Owner control attributes

  skd-glAddOwner CMC-CONTROL ::=
      { GLOwnerAdministration IDENTIFIED BY id-skd-glAddOwner }
  id-skd-glAddOwner OBJECT IDENTIFIER ::= { id-skd 6}

  skd-glRemoveOwner CMC-CONTROL ::=
      { GLOwnerAdministration IDENTIFIED BY id-skd-glRemoveOwner }

  id-skd-glRemoveOwner OBJECT IDENTIFIER ::= { id-skd 7}

  GLOwnerAdministration ::= SEQUENCE {
      glName       GeneralName,
      glOwnerInfo  GLOwnerInfo
  }

  -- This defines the GL Key Compromise control attribute.
  -- It has the simple type GeneralName.

  skd-glKeyCompromise CMC-CONTROL ::=
      { GLKCompromise IDENTIFIED BY id-skd-glKeyCompromise }

  id-skd-glKeyCompromise OBJECT IDENTIFIER ::= { id-skd 8}
  GLKCompromise ::= GeneralName

  -- This defines the GL Key Refresh control attribute.

  skd-glkRefresh CMC-CONTROL ::=
      { GLKRefresh IDENTIFIED BY id-skd-glkRefresh }

  id-skd-glkRefresh OBJECT IDENTIFIER ::= { id-skd 9}

  GLKRefresh ::= SEQUENCE {
      glName  GeneralName,
      dates   SEQUENCE SIZE (1..MAX) OF Date
  }

  Date ::= SEQUENCE {
      start GeneralizedTime,
      end   GeneralizedTime OPTIONAL
  }
  -- This defines the GLA Query Request control attribute.

  skd-glaQueryRequest CMC-CONTROL ::=
      { GLAQueryRequest IDENTIFIED BY id-skd-glaQueryRequest }

  id-skd-glaQueryRequest OBJECT IDENTIFIER ::= { id-skd 11}

  SKD-QUERY ::= TYPE-IDENTIFIER

  SkdQuerySet SKD-QUERY ::= {skd-AlgRequest, ...}
  GLAQueryRequest ::= SEQUENCE {
      glaRequestType   SKD-QUERY.&id ({SkdQuerySet}),
      glaRequestValue  SKD-QUERY.
                           &Type ({SkdQuerySet}{@glaRequestType})
  }

  -- This defines the GLA Query Response control attribute.

  skd-glaQueryResponse CMC-CONTROL ::=
      { GLAQueryResponse IDENTIFIED BY id-skd-glaQueryResponse }

  id-skd-glaQueryResponse OBJECT IDENTIFIER ::= { id-skd 12}

  SKD-RESPONSE ::= TYPE-IDENTIFIER

  SkdResponseSet SKD-RESPONSE ::= {skd-AlgResponse, ...}

  GLAQueryResponse ::= SEQUENCE {
      glaResponseType   SKD-RESPONSE.
                            &id({SkdResponseSet}),
      glaResponseValue  SKD-RESPONSE.
                            &Type({SkdResponseSet}{@glaResponseType})}

  -- This defines the GLA Request/Response (glaRR) arc for
  -- glaRequestType/glaResponseType.

  id-cmc-glaRR OBJECT IDENTIFIER ::=
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) cmc(7) glaRR(99) }

  -- This defines the Algorithm Request

  skd-AlgRequest SKD-QUERY ::= {
     SKDAlgRequest IDENTIFIED BY id-cmc-gla-skdAlgRequest
  }

  id-cmc-gla-skdAlgRequest OBJECT IDENTIFIER ::= { id-cmc-glaRR 1 }
  SKDAlgRequest ::= NULL
  -- This defines the Algorithm Response

  skd-AlgResponse SKD-RESPONSE ::= {
      SMIMECapability{{SKD-Caps}} IDENTIFIED BY
          id-cmc-gla-skdAlgResponse
  }

  id-cmc-gla-skdAlgResponse OBJECT IDENTIFIER ::= { id-cmc-glaRR 2 }
  -- Note that the response for algorithmSupported request is the
  -- smimeCapabilities attribute as defined in RFC 3851.

  -- This defines the control attribute to request an updated
  -- certificate to the GLA.

  skd-glProvideCert CMC-CONTROL ::=
      { GLManageCert IDENTIFIED BY id-skd-glProvideCert }

  id-skd-glProvideCert OBJECT IDENTIFIER ::= { id-skd 13}

  GLManageCert ::= SEQUENCE {
      glName    GeneralName,
      glMember  GLMember
  }

  -- This defines the control attribute to return an updated
  -- certificate to the GLA.  It has the type GLManageCert.

  skd-glManageCert CMC-CONTROL ::=
      { GLManageCert IDENTIFIED BY id-skd-glManageCert }

  id-skd-glManageCert OBJECT IDENTIFIER ::= { id-skd 14}

  -- This defines the control attribute to distribute the GL shared
  -- KEK.

  skd-glKey CMC-CONTROL ::=
      { GLKey IDENTIFIED BY id-skd-glKey }

  id-skd-glKey OBJECT IDENTIFIER ::= { id-skd 15}

  GLKey ::= SEQUENCE {
      glName        GeneralName,
      glIdentifier  KEKIdentifier,   -- See RFC 3852
      glkWrapped    RecipientInfos,  -- See RFC 3852
      glkAlgorithm  KeyWrapAlgorithm,
      glkNotBefore  GeneralizedTime,
      glkNotAfter   GeneralizedTime
  }
  -- This defines the CMC error types

  skd-ExtendedFailures EXTENDED-FAILURE-INFO ::= {
     SKDFailInfo IDENTIFIED BY id-cet-skdFailInfo
  }

  id-cet-skdFailInfo  OBJECT IDENTIFIER ::=
      { iso(1) identified-organization(3) dod(6) internet(1) security(5)
      mechanisms(5) pkix(7) cet(15) skdFailInfo(1) }

  SKDFailInfo ::= INTEGER {
      unspecified           (0),
      closedGL              (1),
      unsupportedDuration   (2),
      noGLACertificate      (3),
      invalidCert           (4),
      unsupportedAlgorithm  (5),
      noGLONameMatch        (6),
      invalidGLName         (7),
      nameAlreadyInUse      (8),
      noSpam                (9),
      deniedAccess          (10),
      alreadyAMember        (11),
      notAMember            (12),
      alreadyAnOwner        (13),
      notAnOwner            (14) }

  END
```

13. 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.

14. Normative References

[ASN1-2002] ITU-T, "ITU-T Recommendation X.680, X.681, X.682, and X.683", ITU-T X.680, X.681, X.682, and X.683, 2002.

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

[RFC3565] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, July 2003.

[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.

[RFC4998] Gondrom, T., Brandner, R., and U. Pordesch, "Evidence Record Syntax (ERS)", RFC 4998, August 2007.

[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 5083, November 2007.

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

[RFC5275] Turner, S., "CMS Symmetric Key Management and Distribution", RFC 5275, June 2008.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 5652, September 2009.

[RFC5912] Hoffman, P. and J. Schaad, "New ASN.1 Modules for the Public Key Infrastructure using X.509 (PKIX)", RFC 5912, June 2010.

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