Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)

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 some auxiliary ASN.1 modules to conform to the 2008 version of ASN.1; the 1988 ASN.1 modules remain the normative version. 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/rfc6268.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF
Contributions published or made publicly available before November
10, 2008. The person(s) controlling the copyright in some of this
material may not have granted the IETF Trust the right to allow
modifications of such material outside the IETF Standards Process.
Without obtaining an adequate license from the person(s) controlling
the copyright in such materials, this document may not be modified
outside the IETF Standards Process, and derivative works of it may
not be created outside the IETF Standards Process, except to format
it for publication as an RFC or to translate it into languages other
than English.

Table of Contents

1. Introduction ....................................................... 3
   1.1. ASN.1 Updates (2002 to 2008) .............................. 4
   1.2. Requirements Terminology .................................. 4
2. ASN.1 Module RFC 3274 ........................................... 5
3. ASN.1 Module RFC 3779 ............................................ 7
4. ASN.1 Module RFC 6019 ............................................ 10
5. ASN.1 Module RFC 4073 ............................................ 11
6. ASN.1 Module RFC 4231 ............................................ 12
7. ASN.1 Module RFC 4334 ............................................ 15
8. ASN.1 Module RFC 5083 ............................................ 16
9. ASN.1 Module RFC 5652 ............................................ 18
10. ASN.1 Module RFC 5752 .......................................... 29
11. Module Identifiers in ASN.1 ..................................... 30
12. Security Considerations ......................................... 32
13. References ....................................................... 32
   13.1. Normative References ..................................... 32
   13.2. Informative References ................................. 33
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 Secure/Multipurpose Internet Mail Extensions (S/MIME).

In this document we have either changed the syntax to use the 2008 ASN.1 standard, or done some updates from previous conversions. The ASN.1 modules updated came from the following RFCs:

- **RFC 3274**, Compressed Data Content Type for Cryptographic Message Syntax (CMS) [RFC3274].
- **RFC 3779**, X.509 Extensions for IP Addresses and AS Identifiers [RFC3779].
- **RFC 6019**, BinaryTime: An Alternate Format for Representing Date and Time in ASN.1 [RFC6019].
- **RFC 4073**, Protecting Multiple Contents with the Cryptographic Message Syntax (CMS) [RFC4073].
- **RFC 4231**, Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 [RFC4231].
- **RFC 4334**, Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN) [RFC4334].
- **RFC 5083**, Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type [RFC5083].
- **RFC 5652**, Cryptographic Message Syntax (CMS) [RFC5652].
- **RFC 5752**, Multiple Signatures in Cryptographic Message Syntax (CMS) [RFC5752].

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 [RFC5911] and [RFC5912], can stand on their own and do not need to import definitions from anywhere else.
This document does not explicitly update the RFCs from which the 
ASN.1 modules have been extracted. This is because the original 1988 
ASN.1 syntax remains the normative version and the modules in this 
document as well as in [RFC5911] and [RFC5912] are informative (but 
hopefully useful) annexes.

1.1. ASN.1 Updates (2002 to 2008)

The modules defined in this document are compatible with the most 
current ASN.1 specification published in 2008 (see [ASN1-2008]). The 
changes between the 2002 specification and the 2008 specification 
include the creation of additional pre-defined types (DATE, DATE-
TIME, DURATION, NOT-A-NUMBER, OID-IRI, RELATIVE-OID-IRI, TIME, TIME-
OF-DAY) and the ability to define different encoding rules (ENCODING-
CONTROL, INSTRUCTIONS). None of the newly defined tokens are 
currently used in any of the ASN.1 specifications published here.

Information on the changes to ASN.1 between the 1988 and 2002 
versions can be found in [RFC6025].

1.2. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", 
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this 
document are to be interpreted as described in [RFC2119].
We have updated the ASN.1 module associated with this document to be 2008 compliant and to use the set of classes previously defined in [RFC5911].

CompressedDataContent-2010
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
   smime(16) modules(0) id-mod-compressedDataContent(54) }

DEFINITIONS IMPLICIT TAGS ::= BEGIN
IMPORTS
  CMSVersion, ContentSet,
  CONTENT-TYPE
FROM CryptographicMessageSyntax-2010
{ iso(1) member-body(2) us(840) rsadsi(113549)
   pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

  AlgorithmIdentifier{}, SMIME-CAPS, ParamOptions
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)};

-- ContentTypes contains the set of content types that are defined in this module.
-- The contents of ContentTypes should be added to ContentSet defined in [RFC5652]

ContentTypes CONTENT-TYPE ::= {ct-compressedData}

-- SMimeCaps contains the set of S/MIME capabilities that are associated with the algorithms defined in this document.
-- SMimeCaps are added to the SMimeCapsSet defined in [RFC5751] as updated by [RFC5911].

SMimeCaps SMIME-CAPS ::= {cpa-zlibCompress.&smimeCaps, ...}
Define the compressed data content type

c-t-compressedData CONTENT-TYPE ::= {
  TYPE CompressedData IDENTIFIED BY id-c-t-compressedData
}

CompressedData ::= SEQUENCE {
  version CMSVersion (v0), -- Always set to 0
  compressionAlgorithm CompressionAlgorithmIdentifier,
  encapContentInfo EncapsulatedContentInfo
}

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

CompressionAlgorithmIdentifier ::= AlgorithmIdentifier{COMPRESS-ALGORITHM, {CompressAlgorithmSet}}

CompressAlgorithmSet COMPRESS-ALGORITHM ::= {
  cpa-zlibCompress, ...
}

-- Algorithm Identifiers

id-alg-zlibCompress OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 8 }

cpa-zlibCompress COMPRESS-ALGORITHM ::= {
  IDENTIFIER id-alg-zlibCompress
  PARAMS TYPE NULL ARE preferredAbsent
  SMIME-CAPS {IDENTIFIED BY id-alg-zlibCompress}
}

-- Content Type Object Identifiers

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

-- Class defined for compression algorithms
COMPRESS-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]
}

END

3. ASN.1 Module RFC 3779

We have updated the ASN.1 module associated with RFC 3779 to be ASN.1 2008 compliant and to use the set of classes previously defined in [RFC5912].

IPAddrAndASCertExtn-2010 { iso(1) identified-organization(3) dod(6) intern(1) security(5) mechanisms(5) pkix(7) mod(0)
  id-mod-ip-addr-and-as-ident-2(72) }
DEFINITIONS EXPLICIT TAGS ::= BEGIN
  EXPORTS ALL;

  IMPORTS

  -- PKIX specific OIDs and arcs --
  id-pe 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) }

  EXTENSION
  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) }
;
Extensions EXTENSION ::= {
    ext-pe-ipAddrBlocks | ext-pe-autonomousSysIds
}

-- IP Address Delegation Extension OID --

ext-pe-ipAddrBlocks EXTENSION ::= {
    SYNTAX IPAddrBlocks
    IDENTIFIED BY id-pe-ipAddrBlocks
}

id-pe-ipAddrBlocks OBJECT IDENTIFIER ::= { id-pe 7 }

-- IP Address Delegation Extension Syntax --

IPAddrBlocks ::= SEQUENCE OF IPAddressFamily

IPAddressFamily ::= SEQUENCE { -- AFI & opt SAFI --
    addressFamily OCTET STRING (SIZE (2..3)),
    ipAddressChoice IPAddressChoice }

IPAddressChoice ::= CHOICE {
    inherit NULL, -- inherit from issuer --
    addressesOrRanges SEQUENCE OF IPAddressOrRange }

IPAddressOrRange ::= CHOICE {
    addressPrefix IPAddress,
    addressRange IPAddressRange }

IPAddressRange ::= SEQUENCE {
    min IPAddress,
    max IPAddress }

IPAddress ::= BIT STRING
-- Autonomous System Identifier Delegation Extension OID --

ext-pe-autonomousSysIds EXTENSION ::= {
  SYNTAX ASIdentifiers
  IDENTIFIED BY id-pe-autonomousSysIds
}

id-pe-autonomousSysIds OBJECT IDENTIFIER ::= { id-pe 8 }

-- Autonomous System Identifier Delegation Extension Syntax --

ASIdentifiers ::= SEQUENCE {
  asnum [0] ASIdentifierChoice OPTIONAL,
  rdi [1] ASIdentifierChoice OPTIONAL

  (WITH COMPONENTS { ..., asnum PRESENT } |
    WITH COMPONENTS { ..., rdi PRESENT })
}

ASIdentifierChoice ::= CHOICE {
  inherit NULL, -- inherit from issuer --
  asIdsOrRanges SEQUENCE OF ASIdOrRange }

ASIdOrRange ::= CHOICE {
  id ASId,
  range ASRange }

ASRange ::= SEQUENCE {
  min ASId,
  max ASId }

ASId ::= INTEGER

END
4. ASN.1 Module RFC 6019

We have updated the ASN.1 module associated with this document to be 2008 compliant and to use the set of classes previously defined in [RFC5911].

BinarySigningTimeModule-2010
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
  pkcs-9(9) smime(16) modules(0)
  id-mod-binSigningTime-2009(55) }
DEFINITIONS IMPLICIT TAGS ::= 
BEGIN
  IMPORTS

  -- From PKIX-CommonTypes-2009 [RFC5912]

  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) }
;

  -- BinaryTime Definition

  -- BinaryTime contains the number seconds since
  -- midnight Jan 1, 1970 UTC.
  -- Leap seconds are EXCLUDED from the computation.

  BinaryTime ::= INTEGER (0..MAX)

  -- Signing Binary Time Attribute

  -- The binary signing time should be added to
  -- SignedAttributeSet and AuthAttributeSet in CMS [RFC5652]
  -- and to AuthEnvDataAttributeSet in [RFC5083] with the
  -- new modules in this document, RFC 6268.
aa-binarySigningTime ATTRIBUTE ::= {
  TYPE BinarySigningTime
  IDENTIFIED BY id-aa-binarySigningTime }

id-aa-binarySigningTime OBJECT IDENTIFIER ::= { iso(1)
  member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
  smime(16) aa(2) 46 }

BinarySigningTime ::= BinaryTime

END

5. ASN.1 Module RFC 4073

We have updated the ASN.1 module associated with this document to be
2008 compliant and to use the set of classes previously defined in
[RFC5911].

ContentCollectionModule-2010
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
  pkcs-9(9) smime(16) modules(0) id-mod-context-Collect-2009(56) }
DEFINITIONS IMPLICIT TAGS ::= BEGIN
IMPORTS
  -- From CryptographicMessageSyntax-2010 [RFC6268]

  CONTENT-TYPE, ContentInfo
  FROM CryptographicMessageSyntax-2010
  { iso(1) member-body(2) us(840) rsadsi(113549)
    pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

  AttributeSet{}, 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) }
;
--
--  An object set of all content types defined by this module.
--  This is to be added to ContentSet in the CMS module
--

  ContentSet CONTENT-TYPE ::= {
    ct-ContentCollection | ct-ContentWithAttributes, ...
  }

Schaad & Turner               Informational                    [Page 11]
-- Content Collection Content Type and Object Identifier
--

c-t-ContentCollection CONTENT-TYPE ::= {
    TYPE ContentCollection IDENTIFIED BY id-c-t-contentCollection } 

id-c-t-contentCollection OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) ct(1) 19 }

ContentCollection ::= SEQUENCE SIZE (1..MAX) OF ContentInfo

--
-- Content With Attributes Content Type and Object Identifier
--

c-t-ContentWithAttributes CONTENT-TYPE ::= {
    TYPE ContentWithAttributes IDENTIFIED BY id-c-t-contentWithAttrs } 

id-c-t-contentWithAttrs OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) ct(1) 20 }

ContentWithAttributes ::= SEQUENCE {
    content  ContentInfo,
    attrs    SEQUENCE SIZE (1..MAX) OF AttributeSet
        {{ ContentAttributeSet }}
}

ContentAttributeSet ATTRIBUTE ::= { ... }

END

6. ASN.1 Module RFC 4231

RFC 4231 does not contain an ASN.1 module to be updated. We have therefore created an ASN.1 module to represent the ASN.1 that is present in the document. Note that the parameters are defined as expecting a parameter for the algorithm identifiers in this module; this is different from most of the algorithms used in PKIX and S/MIME. There is no concept of being able to truncate the MAC (Message Authentication Code) value in the ASN.1 unlike the XML definitions. This is reflected by not having a minimum MAC length defined in the ASN.1.
HMAC-2010  { iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) mod(0) id-mod-hmac(74) }
DEFINITIONS EXPLICIT TAGS ::= BEGIN

EXPORTS ALL;

IMPORTS

MAC-ALGORITHM, 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)};

--
--  This object set contains all of the MAC algorithms that are defined in this module.
--  One would add it to a constraining set of objects such as the MessageAuthenticationCodeAlgorithmSet in [RFC5652]
--
MessageAuthAlgs MAC-ALGORITHM ::= {
    maca-hMAC-SHA224 |
    maca-hMAC-SHA256 |
    maca-hMAC-SHA384 |
    maca-hMAC-SHA512
}

--
--  This object set contains all of the S/MIME capabilities that have been defined for all the MAC algorithms in this module.
--  One would add this to an object set that is used to restrict S/MIME capabilities such as the SMimeCapsSet variable in RFC 3851 (obsoleted by RFC 5751) as modified in RFC 5911.
--
SMimeCaps SMIME-CAPS ::= {
    maca-hMAC-SHA224.&smimeCaps |
    maca-hMAC-SHA256.&smimeCaps |
    maca-hMAC-SHA384.&smimeCaps |
    maca-hMAC-SHA512.&smimeCaps
}

--
--  Define the base OID for the algorithm identifiers
--
rsadsi OBJECT IDENTIFIER ::=  
    {iso(1) member-body(2) us(840) rsadsi(113549)}

digestAlgorithm OBJECT IDENTIFIER ::= {rsadsi 2}  

--  Define the necessary algorithm identifiers  
--

id-hmacWithSHA224 OBJECT IDENTIFIER ::= {digestAlgorithm 8}
id-hmacWithSHA256 OBJECT IDENTIFIER ::= {digestAlgorithm 9}
id-hmacWithSHA384 OBJECT IDENTIFIER ::= {digestAlgorithm 10}
id-hmacWithSHA512 OBJECT IDENTIFIER ::= {digestAlgorithm 11}

--  Define each of the MAC-ALGORITHM objects to describe the  
--    algorithms defined  
--

maca-hMAC-SHA224 MAC-ALGORITHM ::= {  
    IDENTIFIER id-hmacWithSHA224  
    PARAMS TYPE NULL ARE preferredPresent  
    IS-KEYED-MAC TRUE  
    SMIME-CAPS {IDENTIFIED BY id-hmacWithSHA224}
}

maca-hMAC-SHA256 MAC-ALGORITHM ::= {  
    IDENTIFIER id-hmacWithSHA256  
    PARAMS TYPE NULL ARE preferredPresent  
    IS-KEYED-MAC TRUE  
    SMIME-CAPS {IDENTIFIED BY id-hmacWithSHA256}
}

maca-hMAC-SHA384 MAC-ALGORITHM ::= {  
    IDENTIFIER id-hmacWithSHA384  
    PARAMS TYPE NULL ARE preferredPresent  
    IS-KEYED-MAC TRUE  
    SMIME-CAPS {IDENTIFIED BY id-hmacWithSHA384}
}
maca-hMAC-SHA512 MAC-ALGORITHM ::= {
  IDENTIFIER id-hmacWithSHA512
  PARAMS TYPE NULL ARE preferredPresent
  IS-KEYED-MAC TRUE
  SMIME-CAPS {IDENTIFIED BY id-hmacWithSHA512}
}
id-kp-eapOverPPP OBJECT IDENTIFIER ::= { id-kp 13 }
id-kp-eapOverLAN OBJECT IDENTIFIER ::= { id-kp 14 }

-- Wireless LAN SSID Extension

ext-pe-wlanSSID EXTENSION ::= {
  SYNTAX SSIDList
  IDENTIFIED BY id-pe-wlanSSID
  CRITICALITY {FALSE}
}
id-pe-wlanSSID OBJECT IDENTIFIER ::= { id-pe 13 }
SSIDList ::= SEQUENCE SIZE (1..MAX) OF SSID
SSID ::= OCTET STRING (SIZE (1..32))

-- Wireless LAN SSID Attribute Certificate Attribute
-- Uses same syntax as the certificate extension: SSIDList

at-aca-wlanSSID ATTRIBUTE ::= {
  TYPE SSIDList
  IDENTIFIED BY id-aca-wlanSSID
}
id-aca-wlanSSID OBJECT IDENTIFIER ::= { id-aca 7 }

8. ASN.1 Module RFC 5083

This module is updated from RFC 5911 [RFC5911] by the following changes:

1. Define separate attribute sets for the unprotected attributes
   used in EnvelopedData, EncryptedData, and
   AuthenticatedEnvelopedData (RFC 5083).

2. Define a parameterized type EncryptedContentInfoType so that the
   basic type can be used with different algorithm sets (used for
   EnvelopedData, EncryptedData, and AuthenticatedEnvelopedData (RFC
The parameterized type is assigned to an unparameterized type of EncryptedContentInfo to minimize the output changes from previous versions.

Protocol designers can make use of the ‘08 ASN.1 constraints to define different sets of attributes for EncryptedData and EnvelopedData and for AuthenticatedData and AuthEnvelopedData. Previously, attributes could only be constrained based on whether they were in the clear or unauthenticated not on the encapsulating content type.

```
CMS-AudioEncryptedData-2010
  (iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
   smime(16) modules(0) id-mod-cmsAudioEncryptedData-2009(57) )
```

```
DEFINITIONS IMPLICIT TAGS ::= BEGIN
IMPORTS

CMSVersion, EncryptedContentInfoType{},
   MessageAuthenticationCode, OriginatorInfo, RecipientInfos,
CONTENT-TYPE, Attributes{}, ATTRIBUTE, CONTENT-ENCRYPTION,
AlgorithmIdentifier{},
  aa-signingTime, aa-messageDigest, aa-contentType
FROM CryptographicMessageSyntax-2010
  (iso(1) member-body(2) us(840) rsadsi(113549)
   pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) )

ContentEncryptionAlgs
FROM 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) )
;

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

tc-authEncryptedData CONTENT-TYPE ::= {
  TYPE AuthEncryptedData IDENTIFIED BY id-tc-authEncryptedData
}

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

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

EncryptedContentInfo ::= 
  EncryptedContentInfoType { AuthContentEncryptionAlgorithmIdentifier }

AuthContentEncryptionAlgorithmIdentifier ::= AlgorithmIdentifier
  (CONTENT-ENCRYPTION, {AuthContentEncryptionAlgorithmSet})

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

AuthAttributes ::= Attributes{{AuthEnvDataAttributeSet}}

UnauthAttributes ::= Attributes{{UnauthEnvDataAttributeSet}}

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

UnauthEnvDataAttributeSet ATTRIBUTE ::= {...}

END

9. ASN.1 Module RFC 5652

This module is updated from RFC 5911 [RFC5911] by the following changes:

1. Define separate attribute sets for the unprotected attributes used in EnvelopedData, EncryptedData, and AuthenticatedEncryptedData (RFC 5083).

2. Define a parameterized type EncryptedContentInfoType so that the basic type can be used with algorithm sets (used for EnvelopedData, EncryptedData, and AuthenticatedEncryptedData (RFC 5083)). The parameterized type is assigned to an unparameterized type of EncryptedContentInfo to minimize the output changes from previous versions.

We are anticipating the definition of attributes that are going to be restricted to the use of only EnvelopedData. We are therefore separating the different attribute sets so that protocol designers that need to do this will be able to define attributes that are used for EnvelopedData, but not for EncryptedData. The same separation is also being applied to AuthenticatedData and AuthEncryptedData.
CryptographicMessageSyntax-2010

{ iso(1) member-body(2) us(840) rsadsi(113549)
  pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) } 

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) identified-organization(3) dod(6) internet(1) 
    security(5) mechanisms(5) pkix(7) id-mod(0) 
    id-mod-v1AttrCert-02(49) } ;
-- Cryptographic Message Syntax

-- The following are used for version numbers using the ASN.1
-- NOTE: The document reference represents where the versioned
--   feature was introduced to the module.
--
--   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 ::= CLASS {
  &id        OBJECT IDENTIFIER UNIQUE,
  &Type      OPTIONAL
} WITH SYNTAX {
  [TYPE &Type] IDENTIFIED BY &id
}

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,
  encapsContentInfo 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,
  signatureSignatureValue,
  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
    {{ UnprotectedEnvAttributes }} OPTIONAL ]]

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

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

EncryptedContentInfo ::= EncryptedContentInfoType {
  ContentEncryptionAlgorithmIdentifier }

EncryptedContentInfoType {{ AlgorithmIdentifierType }} ::= SEQUENCE {
  contentType CONTENT-TYPE.&id({ContentSet}),
  contentEncryptionAlgorithm AlgorithmIdentifierType,
  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 ) )
}

UnprotectedEnvAttributes ATTRIBUTE ::= { ... }
UnprotectedEncAttributes ATTRIBUTE ::= { ... }

RecipientInfo ::= CHOICE {
  ktri          KeyTransRecipientInfo,
  ...
  [[3: kari   [1] KeyAgreeRecipientInfo ]],
  [[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
    {{UnprotectedEncAttributes}} 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
}

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 ::= { 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 ::= 
    { 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 ::= 
    { 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 ::= 
    { 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 ::= 
    { 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 ::= 
    { 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

10. ASN.1 Module RFC 5752

We have updated the ASN.1 module associated with this document to be
2008 compliant and to use the set of classes previously defined in
[RFC5911].

MultipleSignatures-2010
  { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
    smime(16) modules(0) id-mod-multipleSign-2009(59) }
DEFINITIONS IMPLICIT TAGS ::= 
BEGIN
  -- EXPORTS All
  -- The types and values defined in this module are exported for use
  -- in the other ASN.1 modules. Other applications may use them for
  -- their own purposes.

IMPORTS

  -- Imports from PKIX-Common-Types-2009 [RFC5912]

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) }

  -- Imports from CryptographicMessageSyntax-2010 [RFC6268]

DigestAlgorithmIdentifier, SignatureAlgorithmIdentifier
  FROM CryptographicMessageSyntax-2010
    { iso(1) member-body(2) us(840) rsadsi(113549)
      pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }
11. Module Identifiers in ASN.1

One potential issue that can occur when updating modules is the fact that a large number of modules may need to be updated if they import from a newly updated module. This section addresses one method that can be used to deal with this problem, but the modules in this document don’t currently implement the solution discussed here.
When looking at an import statement, there are three portions: The list of items imported, a textual name for the module, and an object identifier for the module. Full implementations of ASN.1 do module matching using first the object identifier, and if that is not present, the textual name of the module. Note however that some older implementations used the textual name of the module for the purposes of matching. In a full implementation, the name assigned to the module is scoped to the ASN.1 module that it appears in (and thus the need to match the module it is importing from).

One can create a module that contains only the module number assignments and import the module assignments from the new module. This means that when a module is replaced, one can replace the previous module, update the module number assignment module, and recompile without having to modify any other modules.

A sample module assignment module would be:

ModuleNumbers
DEFINITIONS TAGS ::= 
BEGIN
  id-mod-CMS ::= { iso(1) member-body(2) us(840) rsadsi(113549)
                     pkcs(1) pkcs-9(9) smime(16) modules(0) 58 }
  id-mod-AlgInfo ::= 
                     (iso(1) identified-organization(3) dod(6) internet(1)
                      security(5) mechanisms(5) pkix(7) id-mod(0)
                      id-mod-algorithmInformation-02(58))
END

This would be used in the following import statement:

IMPORTS
  id-mod-CMS, id-mod-AlgInfo
FROM ModuleNumbers  -- Note it will match on the name since no
                     -- OID is provided
  CMSVersion, EncapsulatedContentInfo, CONTENT-TYPE
FROM CryptographicMessageSyntax-2010
  id-mod-CMS

AlgorithmIdentifier{}, SMIME-CAPS, ParamOptions
FROM AlgorithmInformation-2009 id-mod-AlgInfo
;
12.  Security Considerations

This 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.

13.  References

13.1.  Normative References


13.2. Informative References


Authors’ Addresses

Jim Schaad
Soaring Hawk Consulting
EMail: ietf@augustcellars.com

Sean Turner
IECA, Inc.
3057 Nutley Street, Suite 106
Fairfax, VA  22031
EMail: turners@ieca.com