

## Cryptographic Message Syntax

### Summary

This Recommendation | International Standard provides ASN.1 modules for using Cryptographic Syntax in ITU-R Recommendations. Cryptographic Message Syntax provides data integrity, confidentiality, origin authenticity, and non-repudiation services needed for reliable information exchange and for strong authentication. It also brings together a set of cryptographic key management techniques to support flexible key establishment mechanisms, such as constructive key management, key agreement, key exchange, and password-based encryption. These techniques can be used to prevent fraud, and to protect personally identifiable and other sensitive information from the threats outlined above. This Recommendation | International Standard will support digital signature, encryption, and signcryption techniques based on the public-key technology defined in the ITU-T X.500 series | ISO/IEC 9594 multipart standard. All the standardized encoding rules for ASN.1 are supported.

### Keywords

AuthenticatedData,  
Cryptographic Message Syntax,  
DigestedData,  
EncryptedData,  
EnvelopedData,  
SigncryptedData,  
SignedData

## 1 Scope

This Recommendation | International Standard enhances the existing cryptographic message syntax protocol by adding signcryption techniques and providing a new ASN.1 module which conforms to the latest edition of the ASN.1 standard which can be used with all standardized encoding rules of ASN.1.

## 2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

### 2.1 Identical Recommendations | International Standards

- Recommendation ITU-T X.509 (2016) | ISO/IEC 9594-8:2017, *Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks*.
- Recommendation ITU-T X.680 (2015) | ISO/IEC 8824-1:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.
- Recommendation ITU-T X.681 (2015) | ISO/IEC 8824-2:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.
- Recommendation ITU-T X.682 (2015) | ISO/IEC 8824-3:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification*.
- Recommendation ITU-T X.683 (2015) | ISO/IEC 8824-4:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.

- Recommendation ITU-T X.690 (2015) | ISO/IEC 8825-1:2015, *Information technology – ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*.
- Recommendation ITU-T X.691 (2015) | ISO/IEC 8825-2:2015, *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*.
- Recommendation ITU-T X.693 (2015) | ISO/IEC 8825-4:2015, *Information technology – ASN.1 encoding rules: XML Encoding Rules (XER)*.
- Recommendation ITU-T X.696 (2015) | ISO/IEC 8825-7:2015, *Information technology – Specification of Octet Encoding Rules (OER)*.
- ISO 11568-1:2005 Banking -- Key management (retail) -- Part 1: Principles
- ISO/IEC 29150:2011, *Information technology – Security techniques – Signcryption*

## 2.2 Additional References

- ANSI X9.73 Cryptographic Message Syntax (CMS) - ASN.1 and XML
- IETF RFC 5652 (2009) *Cryptographic Message Syntax (CMS)*
- IETF RFC 5912 (2010) *New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)*
- IETF RFC 6268 (2011) *Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)*

## 3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply:

The following terms are defined in Rec. ITU-T X.509 | ISO/IEC 9594-8:

- a) attribute certificate
- b) certificate
- c) certificate revocation list

The following term is defined in ISO/IEC 29150:2011

signcryption

## 4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

CMS     Cryptographic Message Syntax  
CRL     Certificate Revocation List  
DBEKM Database Encryption Key Management

## 5 Conventions

## 6 Cryptographic Message Syntax

Cryptographic Message Syntax (CMS) has been defined by IETF. The base text is in RFC 5652. ASN.1 modules have been revised to conform to the current ASN.1 standard and have been published in RFC 6268.

CMS defines the following content types:

- data: used to transfer data defined string of octets;
- signed data: used to transfer data with zero or more signatures;

- enveloped data: used to transfer encrypted data with one or more content-encryption keys;
- digested data: used to transfer data with a message digest;
- encrypted data: used to transfer encrypted data;
- authenticated data: used to transfer data with a message authentication code and one or more encrypted authentication keys;

Each of these content types is uniquely identified by an object identifier:

- for data:

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

- for signed data:

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

- for enveloped data:

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

- for digested data:

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

- for encrypted data:

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

- for authenticated data:

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

Data transferred with Cryptographic Message Syntax use the following ASN.1 type:

```
ContentInfo ::= SEQUENCE {
    contentType CONTENT-TYPE.&id({ContentSet}),
    content      [0] EXPLICIT CONTENT-TYPE.&Type({ContentSet}{@contentType})}
```

The CONTENT-TYPE information object class is defined as TYPE-IDENTIFIER and is used to assign one of the previous object identifiers to the corresponding ASN.1 type.

```
CONTENT-TYPE ::= TYPE-IDENTIFIER
```

```
ContentType ::= CONTENT-TYPE.&id
```

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

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

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

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

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

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

```
ct-AuthenticatedData CONTENT-TYPE ::= {AuthenticatedData IDENTIFIED BY
```

**id-ct-authData}**

Other content types can be defined by creation of new information objects of **CONTENT-TYPE** information object class using unique object identifiers.

The **ct-SignedCryptedData** defined in clause 7 is an example.

## 7 Signcryption

The **SigncryptedData** uses the signcryption technique defined in ISO/IEC 29150. The signcryption technique simultaneously signs and encrypts the data to achieve origin authentication, data integrity and confidentiality. Signcryption can be used in CMS in four different modes:

- a) **signcrypted-content**: content of any type or format is signcrypted using the signcryption algorithm;
- b) **signcrypted-attributes**: content of any type or format and a collection of attributes of any type or format are together signcrypted;
- c) **signcrypted-components**: elements of content of any type or format are signcrypted for one or more message recipients using the public-private keys of the sender and the public key of each recipient.
- d) **signcrypted-envelope**: a fresh symmetric key is used to encrypt content of any type or format and the resulting ciphertext is transmitted as an octet string.

### 7.1 The SigncryptedData type

```
id-signcryptedData OBJECT IDENTIFIER ::=
    {itu-t recommendation(0) x(24) cms-profile(894) signcryption(0) data(0)}
```

```
ct-SigncryptedData CONTENT-TYPE ::= {
    TYPE SigncryptedData IDENTIFIED BY id-signcryptedData}
```

```
SigncryptedData ::= SEQUENCE {
    version          CMSVersion,
    contentInformation ContentInformation,
    certificates     CertificateSet OPTIONAL,
    crls             RevocationInfoChoices OPTIONAL,
    signcrypters    Signcrypters
}
```

```
Signcrypters ::= SEQUENCE SIZE(1..MAX) OF Signcrypter
```

- a) **version** is version number.
- b) **contentInformation** identifies the signcrypted data processing mode.
- c) **certificates** is a collection of public key and/or attribute certificates to facilitate the validation of the public keys of the signcrypters. This collection may contain more certificates than necessary or fewer and, in the latter case, recipients have to use other means to get other certificates.
- d) **crls** is a collection of public key and/or attribute certificate revocation lists to facilitate the validation of the certificates of the signcrypters. As for **certificates**, this collection may contain more certificate revocation lists than necessary or fewer and, in the latter case, recipients have to use other means to get other certificate revocation lists.
- e) **signcrypters** is a collection of all signcrypter specific information.

### 7.2 The ContentInformation type

```
ContentInformation ::= SEQUENCE {
    mode          Mode,
    content      Content OPTIONAL
}
```

The modes are defined using the following information object class:

```
MODE ::= CLASS {
    &Type          OPTIONAL,
    &id            OBJECT IDENTIFIER UNIQUE
}
```

WITH SYNTAX {[WITH SYNTAX &Type] ID &id}

Four modes are currently defined:

- a) **signcryptedExceptions**: content and attributes of any type or format are signcryptedExceptions.
- b) **signcryptedExceptions**: specified components of content of any type or format are signcryptedExceptions, and the resulting cryptogram is signed along with a manifest of signcryptedExceptions locations and other attributes.
- c) **signcryptedExceptions**: content of any type or format is signcryptedExceptions.
- d) **signcryptedExceptions**: content of any type or format is encrypted under a symmetric key to create ciphertext for sharing with one or more message recipients. That symmetric key and a message digest of the ciphertext are signcryptedExceptions for each message recipient using the public-private keys of the sender and the public key of each recipient.

Mode ::= MODE.&id({ProcessingModes})

```
ProcessingModes    MODE ::= {
    signcryptedExceptions    |
    signcryptedExceptions    |
    signcryptedExceptions    |
    signcryptedExceptions,
    ... -- Expect additional processing modes --
}
```

Content ::= OCTET STRING(SIZE(1..MAX))

### 7.2.1 The signcryptedExceptions mode

In the signcryptedExceptions-content mode, content of any type or format is signcryptedExceptions using the signcryptedExceptions algorithm identified in the **signcryptedExceptionsDataAlgorithm** component of the **signcryptedExceptions** component of type **SigncryptedExceptionsData**. The message sender applies this signcryptedExceptions algorithm to the content using the sender public and private keys and the recipient public key. These keys are identified in the message as values of type **SigncryptedExceptionsIDs** and the **signcryptedExceptionsValue** component of type **SigncryptedExceptions** contains the results of signcryptedExceptions.

In this processing mode of **SigncryptedExceptionsData**, there are no signcryptedExceptions attributes and the optional **signatureInformation** component is not present. The message sender at their option may include values in the **unsigncryptedExceptionsAttributes** component of the the **SigncryptedExceptions** component for any recipient. The signcryptedExceptions-content mode is indicated in a message by the following information object identifier:

**signcryptedExceptionsContent** MODE ::= { ID **signcryptedExceptions-content** }

A value of type **SigncryptedExceptionsData** using signcryptedExceptions-content mode is created as follows:

- a. The value of the **version** component of type **SigncryptedExceptionsData** is set to 1.
- b. In the **contentInformation** component of type **SigncryptedExceptionsData**, set the value of mode to the signcryptedExceptions-content object identifier. The optional **content** component should not be present.
- c. Optionally, include values in the **certificates** component of type **SigncryptedExceptionsData**.
- d. Optionally, include values in the **crls** component of type **SigncryptedExceptionsData**.
- e. For each message recipient, include a value of type **SigncryptedExceptions** in the **signcryptedExceptions** component of type **SigncryptedExceptionsData** as follows:
  - a. The value of the **version** component of type **SigncryptedExceptions** is set to 1.
  - b. In the **sids** component of type **SigncryptedExceptions** set the components of a value of type **SigncryptedExceptionsIDs** to identify the public-private key pairs of the message sender and recipient.
  - c. Set the value of the **signcryptedExceptionsDataAlgorithm** component of type **SigncryptedExceptions** to identify the signcryptedExceptions algorithm and any associated parameters.
  - d. Include the results of **signcryptedExceptions** content of any type or format encoded as a value of type OCTET STRING in the **signcryptedExceptionsValue** component of type **SigncryptedExceptions**.
  - e. The optional **signatureInformation** component of type **SigncryptedExceptions** should not be present.

- f. Optionally, include values in the **unsignedAttributes** component of type **Signcrypter**.

To recover the plaintext from the **SigncryptData** message, a message recipient should perform the following steps:

1. Search the list of per recipient values of type **Signcrypter** in the **signcrypters** component of type **SigncryptData** to locate the recipient public-private key pair in the **sids** component of type **Signcrypter**.
2. Decrypt the value in the **signcryptValue** component of type **Signcrypter** for a given recipient using their public-private key pair of the recipient and the public key of the sender identified in the **sids** component of type **Signcrypter**, and the signcrypt algorithm and associated parameters provided in the **signcryptDataAlgorithm** component of type **Signcrypter**.

Perform certificate path validation to gain assurance that the sender public key certificate is trusted.

### 7.2.2 The **signcryptAttributes** mode

In the signcrypt-attributes mode, content of any type or format and a collection of attributes of any type or format in the **signcryptAttributes** component of type **SigncrypterInfo** are together signcrypt as specified for the signcrypt-content mode. At least three attributes shall be present: the **messageDigest**, the **contentType**, and the **signcryptAttributes** attribute.

The signcrypt-attributes mode is indicated in a message by the following information object identifier:

**signcryptAttributes** MODE ::= { ID **signcrypt-attributes** }

A value of type **SigncryptData** using signcrypt-attributes mode is created as follows:

- a) The value of the **version** component of type **SigncryptData** is set to 1.
- b) In the **contentInformation** component of type **SigncryptData**, set the value of mode to the **signcrypt-attributes** object identifier. The optional **content** component should not be present.
- c) Optionally, include values in the **certificates** component of type **SigncryptData**.
- d) Optionally, include values in the **crls** component of type **SigncryptData**.
- e) For each message recipient, include a value of type **Signcrypter** in the **signcrypters** component of type **SigncryptData** as follows:
  - i. The value of the **version** component of type **Signcrypter** is set to 1.
  - ii. In the **sids** component of type **Signcrypter** set the components of a value of type **SigncrypterIDs** to identify the public-private key pairs of the message sender and recipient.
  - iii. Set the value of the **signcryptDataAlgorithm** component of type **Signcrypter** to identify the signcrypt algorithm and any associated parameters.
  - iv. Create a value of type **ToBeSigncrypt** by setting its **content** component to a value of type **Content** and its **attributes** component to a value of type **SigncryptAttributes**.
  - v. Signcrypt an encoded value of type **ToBeSigncrypt** and include the results of the processing in the **signcryptValue** component of type **Signcrypter**.
  - vi. The optional **signatureInformation** component of type **SigncrypterInfo** should not be present.
  - vii. Optionally, include values in the **unsignedAttributes** component of type **Signcrypter**.

To recover the plaintext from the **SigncryptData** message, a message recipient should perform the following steps:

- a) Search the list of per recipient values of type **Signcrypter** in the **signcrypters** component of type **SigncryptData** to locate the recipient public-private key pair in the **sids** component of type **Signcrypter**.
- b) Decrypt the value in the **signcryptValue** component of type **Signcrypter** for the recipient using the signcrypt algorithm and any associated parameters provided in the **signcryptDataAlgorithm** component of type **Signcrypter** to recover a value of type **ToBeSigncrypt**.
- c) Perform certificate path validation to gain assurance that the sender public key certificate is trusted.

### 7.2.3 The `signcryptComponents` mode

In `signcrypt-components` mode elements of content of any type or format are `signcrypt`d for one or more message recipients using the public-private keys of the sender and the public key of each recipient. A content-type specific manifest of `signcrypt`d element locations in the content are bound to the partially `signcrypt`d content under a digital signature.

The `signcrypt-components` mode is indicated in a message by the following information object identifier:

```
signcryptComponents MODE ::= { ID signcrypt-components }
```

A value of type `SigncryptPartsManifest` is defined in terms of the `&id` and `&Type` fields of the `SIGCRYPTED` information object class. This type definition uses a parametrized type, `Signcrypted{}`, whose sole parameter is the `Manifest` information object set:

```
SigncryptPartsManifest ::= Signcrypted({Manifest})

Signcrypted{SIGCRYPTED:IOSet} ::= SEQUENCE {
    Name          SIGCRYPTED.&id({IOSet}),
    Parts         SIGCRYPTED.&Type({IOSet}{@name}) OPTIONAL
}

SIGCRYPTED ::= CLASS {
    &id    OBJECT IDENTIFIER UNIQUE,
    &Type  OPTIONAL
}
WITH SYNTAX {OID &id [PARMS &Type]}

Manifest SIGCRYPTED ::= {
    XPathManifest,
    ...
}
```

The type of manifest required to identify the `signcrypt`d components of a particular object varies with the type, structure, and format of the object. For content in the form of an image, a `signcrypt`d components manifest might be composed of a list of ((x, y), (x, y)) coordinate pairs that define a rectangular area of the image to be `signcrypt`d and thereby redacted.

In this standard, a single manifest object is defined to support component identification in an XML document. The `xPathManifest` object uses location paths based on the XPath query language to specify a set of one or more `signcrypt`d XML document components and is defined as follows:

```
xPathManifest SIGCRYPTED ::= {
    OID id-cms-XPath  PARMS XPathSet
}

XPathSet ::= SEQUENCE SIZE(1..MAX) OF XPath

XPath ::= UTF8String (CONSTRAINED BY { -- XML Path Query Language 2.0 -- })
```

A value of type `SigncryptData` using `signcrypt-components` mode is created as follows:

- a) The value of the `version` component of type `SigncryptData` is set to 1.
- b) In the `contentInformation` component of type `SigncryptData`, set the value of `mode` to the `signcrypt-components` object identifier. The optional `content` component should not be present.
- c) Optionally, include values in the `certificates` component of type `SigncryptData`.
- d) Optionally, include values in the `crls` component of type `SigncryptData`.
- e) For each message recipient, include a value of type `Signcrypter` in the `signcrypters` component of type `SigncryptData` as follows:
  - i. The value of the `version` component of type `Signcrypter` is set to 1.
  - ii. In the `sids` component of type `Signcrypter` set the components of a value of type `SigncrypterIDs` to identify the public-private key pairs of the message sender and recipient.
  - iii. `Signcrypt` one or more elements of the content and include the partially `signcrypt`d results as the value of the `signcryptValue` component of type `Signcrypter`.

- iv. In the **signatureInformation** component of type **Signcrypter**, identify the signing key in the optional **signerIdentifier** component if this key differs from the key used to signcrypt.
- v. If the signature algorithm is not a system default or is not indicated in the signcrypted data algorithm identifier, identify the signature algorithm object identifier and any associated parameters in the **signatureAlgorithm** component of type **Signcrypter**.
- vi. Prepare a manifest value in the **signcryptedExceptionManifest** component of type **ToBeSigned** that indicates the location of each signcrypted element in the **signcryptedExceptionValue** component of type **Signcrypter**.
- vii. Include the **contentType** and **messageDigest** attributes in the **signedAttributes** component of type **ToBeSigned**.
- viii. Place the results of signing a value of type **ToBeSigned** in the **signatureValue** component of type **SignatureInformation**.
- ix. Optionally, include values in the **unsigncryptedExceptionAttributes** component of type **Signcrypter**.

To recover the plaintext from the **SigncryptedExceptionData** message, a message recipient should perform the following steps for each message recipient:

- a. Search the list of per recipient values of type **Signcrypter** in the **signcrypters** component of type **SigncryptedExceptionData** to locate the recipient public-private key pair in the **sids** component of type **Signcrypter**.
- b. In the **signatureInformation** component of type **Signcrypter**, verify the signature in the **signatureValue** component over a value of type **ToBeSigned** using the signcryptedException algorithm and any associated parameters provided in the **signcryptedExceptionAlgorithm** component of type **Signcrypter**.
- c. Using the manifest in the **signcryptedExceptionManifest** component of type **ToBeSigned**, designcrypt each of the indicated signcryptedException elements listed in the manifest to recover the partially signcryptedException components of the content.
- d. Perform certificate path validation to gain assurance that the sender certificates are trusted.

#### 7.2.4 The signcryptedExceptionEnvelope mode

In the signcryptedException-envelope mode, a fresh symmetric key is used to encrypt content of any type or format, and the resulting ciphertext is placed in the **encryptedContentInfo** component of a value of type **NamedKeyEncryptedData**, a type defined in X9.73 that may include an optional key name and optional extensions.

```
NamedKeyEncryptedData ::= SEQUENCE {  
    Version                CMSVersion,  
    keyName                [0] OCTET STRING OPTIONAL,  
    encryptedContentInfo   EncryptedContentInfo,  
    unprotectedAttrs      [1] UnprotectedEncAttributes OPTIONAL  
}
```

The **EncryptedContentInfo** type is defined in CMS protocol as:

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

The **encryptedContentInfo** component of **NamedKeyEncryptedData** type also contains the required algorithm object identifier and any associated parameters of the symmetric encryption algorithm used to encrypt the content. This prepared value of type **NamedKeyEncryptedData** is then encoded as an **OCTET STRING** and placed in the **content** component of the **contentInfo** component of type **SigncryptedExceptionData**. The **contentType** component of type **EncryptedContentInfo** is set to the information object identifier value signcryptedException-envelope.

The **SigncryptedExceptionData** message contains one value of type **Signcrypter** for each message recipient. The **Signcrypter** type contains a **signcryptedExceptionAttributes** component that must contain the symmetric encryption key signcryptedException for each message recipient in a **signcryptedExceptionEnvelope** attribute.



The **signcryptedExceptionEnvelope** attribute is defined as follows:

```
signcryptedExceptionEnvelope ATTRIBUTE ::= {  
    WITH SYNTAX SigncryptedExceptionKey ID signcryptedExceptionEnvelope  
}
```

```
SigncryptedExceptionKey ::= OCTET STRING
```

When the **signcryptedExceptionEnvelope** attribute is included in the **signcryptedExceptionAttributes** component of type **SigncryptedExceptioner**, both the **messageDigest** and **contentType** attributes shall be present. To complete construction of the message, the **eContent** is concatenated together with all **signcryptedExceptionAttributes** and signed by the message sender. The completed message can then be transmitted to all recipients.

On receiving the message, a recipient first verifies the signature on the message. To recover the payload encrypted in the **signcryptedExceptionEnvelope** attribute, a message recipient must first designdecrypt the content in the value of type **EncapsulatedContentInfo** to recover the symmetric key, then use the recovered key to decrypt the payload.

A value of type **SigncryptedExceptionData** using signcryptedException-envelope mode is created as follows:

- a. The value of the **version** component of type **SigncryptedExceptionData** is set to 1.
- b. In the **contentInformation** component of type **SigncryptedExceptionData**, set the value of mode to the signcryptedException-envelope object identifier.
- c. Using a fresh symmetric key, encrypt content of any type or format and place the encrypted content in the optional **encryptedContentInfo** component of a value of type **NamedKeyEncryptedData**.
- d. Complete preparation of a value of type **NamedKeyEncryptedData**, by optionally providing a name for the symmetric key in the **keyName** component, and identifying the symmetric content encryption algorithm and any parameters in the **encryptedContent** component.
- e. Embed the prepared value of type **NamedKeyEncryptedData** in the optional content component in the **contentInformation** component of type **SigncryptedExceptionData**.
- f. Prepare a **messageDigest** attribute that identifies a message digest algorithm and contains a digest of the **encryptedContent** component of type **NamedKeyEncryptedData**.
- g. Prepare a value of type **ToBeSigncryptedException** that includes this **messageDigest** attribute in the **attributes** component, and the symmetric content encryption key in the **contents** component, a value of type **ToBeSigncryptedException** that will be signcryptedException for each message recipient.
- h. Optionally, include values in the **certificates** component of type **SigncryptedExceptionData**.
- i. Optionally, include values in the **crls** component of type **SigncryptedExceptionData**.
- j. For each message recipient, include a value of type **SigncryptedExceptioner** in the **signcryptedExceptioners** component of type **SigncryptedExceptionData** as follows:
  - i. The value of the **version** component of type **SigncryptedExceptioner** is set to 1.
  - ii. In the **sids** component of type **SigncryptedExceptioner** set the components of a value of type **SigncryptedExceptionerIDs** to identify the public-private key pairs of the message sender and recipient.
  - iii. Set the value of the **signcryptedExceptionDataAlgorithm** component of type **SigncryptedExceptioner** to identify the signcryptedException algorithm and any associated parameters.
  - iv. Include the results of signcryptedException the prepared value of type **ToBeSigncryptedException** in the **signcryptedExceptionValue** component of type **SigncryptedExceptioner**.
  - v. The optional **signatureInformation** component of type **SigncryptedExceptioner** should not be present.
  - vi. Optionally, include values in the **unsigncryptedExceptionAttributes** component of type **SigncryptedExceptioner**.

To recover the plaintext from the **SigncryptedExceptionData** message, a message recipient should perform the following steps:

- a. Search the list of per recipient values of type **SigncryptedExceptioner** in the **signcryptedExceptioners** component of type **SigncryptedExceptionData** to locate the recipient public-private key pair in the **sids** component of type **SigncryptedExceptioner**.

- b. Decode the optional **content** component in the **contentInformation** component of type **SigncrypteData**, to recover a value of type **NamedKeyEncryptedData** from the octet string.
- c. Designcrypt the value in the **signcryptValue** component of type **Signcrypter** for a given recipient using their public-private key pair of the recipient and the public key of the sender to recover the signcrypte value of type **ToBeSigncrypte** that contains a symmetric key and a message digest of the content encrypted using the symmetric key.
- d. Perform certificate path validation to gain assurance that the sender public key certificate is trusted.
- e. Verify that the symmetric key encrypted content has not been modified by computing a digest of the **encryptedContentInfo** component of type **NamedKeyEncryptedData**. using the algorithm in the message digest attribute recovered from the signcrypte value of type **ToBeSigncrypte** and comparing the two digests for equivalency.
- f. Use the symmetric key recovered from designcrypting the value of type **ToBeSigncrypte** to decrypt the content value of the **contentInformation** component of type **SigncrypteData**.

### 7.3 The Signcrypter type

```
Signcrypter ::= SEQUENCE {  
    Version                Version,  
    Sids                   SigncrypterIDs,  
    signcrypteDataAlgorithm SigncrypteDataAlgorithmIdentifier,  
    signcryptValue         SigncryptValue,  
    signatureInformation   SignatureInformation OPTIONAL,  
    unsigncryptAttributes UnSigncryptAttributes OPTIONAL  
}
```

The public-private key pairs of the message sender and recipient are identified as they are in the **SigncrypteData** message as values of type **SigncrypterIDs** defined as follows:

```
SigncrypterIDs ::= SEQUENCE {  
    Sender    KeyPairIdentifier,  
    Recipient KeyPairIdentifier  
}
```

**KeyPairIdentifier ::= SignerIdentifier**

A message sender uses their own public and private keys along with the public key of the recipient to signcrypt content. The message recipient uses the public key of the sender with their own public and private keys to designcrypt the content signcrypte by the sender.

The **SigncrypteDataAlgorithmIdentifier** type contains the signcrypte algorithm identifier and any associated parameters. For all of the signcrypte algorithms defined in the ISO/IEC 29150 standard, the parameters include one of the hash functions and one of the key derivation functions defined in ISO/IEC 18033.

```
SigncrypteDataAlgorithmIdentifier ::=  
    AlgorithmIdentifier {{SigncryptAlgorithms}}  
  
SigncryptAlgorithms ALGORITHM ::= {  
    SigncryptMechanism, -- ISO/IEC 29150 Signcrypt algorithms --  
    ... -- Expect additional algorithm objects --  
}
```

A **signcryptValue** component of type **Signcrypter** contains the results of signcrypte, and is defined as follows:

**SigncryptValue ::= OCTET STRING (SIZE(1..MAX))**

An optional **signatureInformation** component of type **Signcrypter** contains information needed to support the use of a digital signature when the signcrypte-components mode of processing is used. The definition of type **signatureInformation** is provided in the detailed processing description that follows.

```
SignatureInformation ::= SEQUENCE {  
    signerIdentifier      SignerIdentifier OPTIONAL,  
    signatureAlgorithm    SignatureAlgorithmIdentifier OPTIONAL,  
    toBeSigned           ToBeSigned,  
    signatureValue       SignatureValue  
}
```

```
ToBeSigned ::= SEQUENCE {
    signcryptePartsManifest SigncryptePartsManifest,
    signedAttributes        SignedAttributes
}
```

```
SigncryptePartsManifest ::= Signcrypte({Manifest})
```

```
Manifest SIGNCRYPTED ::= {
    xPathManifest,
    ... -- Expect additional manifest types --
}
```

```
xPathManifest SIGNCRYPTED ::= {
    OID xPath PARMS XPathSet
}
```

```
XPathSet ::= SEQUENCE (SIZE(1..MAX)) OF XPath
```

```
XPath ::= UTF8String(CONSTRAINED BY { -- XML Path Language 2.0 --})
```

An optional `unsigncrypteAttributes` component of type `Signcrypter` contains any attribute values that may be defined by or required by the message sender. These attributes are not protected by signcrypte data processing. Type `UnsigncrypteAttributes` is defined as follows:

```
UnsigncrypteAttributes ::=
    SEQUENCE SIZE(1..MAX) OF Attribute {{UnsigncrypteAttributes}}
```

```
UnsigncrypteAttributes ATTRIBUTE ::= {
    ... -- Expect additional attributes --
}
```

## 8 Other key management techniques

### 8.1 The Constructive Key Management

The Constructive Key Management (CKM) is a dynamic symmetric key management framework described in X9.69. It is a method of establishing a key, whereby several components of keying material, both symmetric and asymmetric type of keys, where each component is used for a specific purpose, are combined together using a mathematical function to produce an object key (content key).

Internal keys associated with CKM meet criteria established in ISO 11568. The mathematical function inherent in CKM is a two-part process for binding an analog representation like a naming convention to an asymmetric or symmetric type of key and having that part of the process used to provide integrity to a second part which includes a random number and other symmetric key functionality which together results in a content encryption key. The first part of the process establishes labels (or attributes) that define characteristics of protected data through a naming convention to create a cryptographic enforced access control for content (as objects) resulting from the encryption action related to the content encryption key. Protecting the data and access control for the data is enforced by the framework. Data peerage can be done through combinations of logic relationships. Each encryption event results in a new random number. A key recovery capability is inherent in the framework and an administrative capability identified in X9.69.

```
ckmRecipientInfo KEY-MANAGEMENT ::=
    { KeyConstructRecipientInfo IDENTIFIED BY id-ckm-recipient-info }
```

is the information object that is used to populate the components of type `OtherRecipientInfo`. The object identifier value in component `oriType` is the value `id-ckm-recipient-info`, and the associated type is a value in component `oriValue` of type `KeyConstructRecipientInfo` in its encoded format.

In an XML encoding, a CKM value of type `OtherRecipientInfo` is

```
<OtherRecipientInfo>
  <oriType>
    1.2.840.10060.2.1 <!-- id-ckm-recipient-info -->
  </oriType>
  <oriValue>
    <KeyConstructRecipientInfo>
      <version> 1 </version>
      <did>
        <domainName>domainName</domainName>
      </did>
    </KeyConstructRecipientInfo>
  </oriValue>
</OtherRecipientInfo>
```

```
<domainMaintenanceLevel>1</domainMaintenanceLevel>
</did>
<ckmid>
  <unencrypted> 1 2 3</unencrypted>
</ckmid>
<ukm>some-base64-text</ukm>
<keyConstructionAlgorithm>
  <algorithm>algorithmIdentifierOid</algorithm>
</keyConstructionAlgorithm>
<encryptedRandom>some-base64-text</encryptedRandom>
</KeyConstructRecipientInfo>
</oriValue>
<OtherRecipientInfo>
```

In this example, ukm and encryptedRandom values contain octet string values encoded in base 64. The algorithmIdentifier value contains an object identifier value.

## 8.2 Database Encryption Key Management

Database Encryption Key Management (DBEKM) is a key management technique defined in X9.73:2017 for managing symmetric keys in a database environment. This technique relies on the secure exchange of information between a secure cryptographic device (SCD) and a server. The dbekmRecipientInfo object of class KEY-MANAGEMENT is defined as follows:

```
dbekmRecipientInfo KEY-MANAGEMENT ::=
  { DBEKMRecipientInfo IDENTIFIED BY id-dbekm-recv-info }
```

This information object that is used to populate the components of type **OtherRecipientInfo**. The object identifier value in component **oriType** is the value **id-dbekm-recv-info**, and the associated type is a value in component **oriValue** of type **DBEKMRecipientInfo** in its encoded format.

The DBEKM method is designed to support using a data encryption key in cryptographic software and using a master key in cryptographic hardware, also referred to as a secure cryptographic device (SCD) without violating the cryptographic boundary of the SCD or spoofing the SCD by presenting the SCD with a data object that is intended for use as a cryptographic key. Further, DBEKM eliminates the need of relying on a software object using a password-based key derivation function (PBDF) to protect data encryption keys.

The DBEKM method employs three key types:

1. Master Key Encryption Key (MK) used to protect the HMAC Key (HK)
2. Keyed Hash Message Authentication (HMAC) Key (HK) used to generate a seed
3. Data Encryption Key (DK) used to protect data

The DBEKM method employs four cryptographic functions:

1. Algorithms for data and key encryption, i.e. AES
2. Algorithms for Keyed Hash Message Authentication (HMAC)
3. Algorithms for hashing, e.g. SHA2, SHA3, etc.
4. Algorithms for key derivation functions (KDF), i.e. ISO/IEC 11770-6

The DBEKM method has the following benefits:

1. The cryptographic hardware generates the master key (MK) and the HMAC key (HK)
2. The cryptographic hardware retains the MK but not the HK
3. The cryptographic hardware generates a seed per the unique database identifier
4. The cryptographic hardware does not know the KDF and cannot generate the data key (DK)
5. The server does not store any cleartext keys or passwords protecting keys
6. The server manages its own unique database identifier

The DBEKM method has the following security considerations:

1. The seed is securely transmitted from the cryptographic hardware to the database server
2. The server cryptographic software generates a DK using a KDF based on the seed
3. The server memory contains the cleartext data key (DK) for use with cryptographic software

The DBEKM method can be used for generating a DK as a CEK, for changing a DK, using multiple DK with a single server when protecting different data elements, and using multiple DK or MK with multiple servers.

### **8.2.1 Single server initial data encryption key**

For a single server initial data encryption key, a secure connection is first established between the SCD and the server storage. The SCD generates the MK and HK, encrypts the HK using the MK, and sends the HK cryptogram to the server over a secured channel. The server stores the HK cryptogram and at some previous or subsequent point in time generates a unique identifier (ID) that represents a meaningful database name. Meanwhile, the SCD destroys the HK but retains the MK.

To obtain a seed for the DK the server sends a request to the SCD containing the HK cryptogram and the unique ID over a secured channel. The SCD decrypts the HK using the MK, generates a seed using the HMAC algorithm with the HK and the unique ID, and sends the seed to the server over a secured channel. The server generates the DK using a KDF with the seed and installs the DK into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and the seed.

### **8.2.2 Single server data encryption key change**

For a single server data encryption key change, a secure connection is first established between the SCD and the server storage. The server can change the DK at any time by changing the unique ID. The data was previously encrypted by sending a request to the SCD with the HK cryptogram and unique ID, generating the DK using the KDF with the seed received from the SCD, and encrypting the data using the DK in memory.

To change the DK the server decrypts the data, generates and stores a new ID, sends a request to the SCD with the HK cryptogram and the new ID, generates a new DK using the new seed, and encrypts the data using the new DK in memory. The server can decrypt with the old DK and re-encrypt with the new DK all of the data at once, or manage a gradual migration between the two DK.

The SCD only retains the MK, it destroys the old and new DK, the old and new ID, and the old and new seeds. As long as the server can manage its IDs it can manage its DKs accordingly but it cannot generate the DK without obtaining the seed from the SCD. The server can change the DK at any time by managing its ID, and if needed can recover old DK as long as it archives the HK cryptogram and the associated ID.

### **8.2.3 Multiple data encryption keys with single server**

A single server might need multiple encryption keys to protect different information on a single server instead of using the same key for all data that needs encryption. The server can manage multiple DKs by managing multiple IDs. The SCD generates a single MK and single HK, encrypts the HK using the MK, and sends the HK cryptogram to the server over a secured channel. The server stores the HK cryptogram and at some previous or subsequent point in time the server generates multiple unique identifiers (IDs), one for each DK. Meanwhile, the SCD destroys the HK but retains the MK.

To obtain a seed s1 for the first DK1 the server sends a request to the SCD containing the HK cryptogram and the first unique ID1 over a secured channel. The SCD decrypts the HK using the MK, generates seed s1 using the HMAC algorithm with the HK and the unique ID1, and sends the seed s1 to the server over a secured channel. The server generates the first DK1 using a KDF with the seed s1 and installs the DK1 into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and the seed s1.

To obtain a seed s2 for the second DK2 the server sends a request to the SCD containing the HK cryptogram and the second unique ID2 over a secured channel. The SCD decrypts the HK using the MK, generates seed s2 using the HMAC algorithm with the HK and the unique ID2, and sends the seed s2 to the server over a secured channel. The server generates the second DK2 using a KDF with the seed s2 and installs the DK2 into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and the seed s2. When the server is restarted and the DKs are erased from memory, the server can regenerate the DKs by reacquiring the seeds at any time by resending the same HK cryptogram and unique IDs to the SCD.

### **8.2.4 Multiple data encryption keys with multiple servers**

Multiple servers collocated within the same data-center might need a unique encryption key per server instead of sharing the same key on all servers. A secure connection is first established between the SCD and each of the storage of each server. The SCD generates a single MK and single HK, encrypts the HK using the MK, and sends the HK cryptogram to both servers over a secured channel. The servers store the HK cryptogram and at some previous or subsequent point in time each server generates a unique identifier, ID1 for server S1 and ID2 for server S2. Meanwhile, the SCD destroys the HK but retains the MK.

To obtain a seed s1 for DK1 the server S1 sends a request to the SCD containing the HK cryptogram and the unique ID1 over a secured channel. The SCD decrypts the HK using the MK, generates seed s1 using the HMAC algorithm with the

HK and the unique ID1, and sends the seed s1 to server S1 over a secured channel. The server S1 generates DK1 using a KDF with the seed s1 and installs the DK1 into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and the seed s1.

To obtain a seed s2 for DK2 the server S2 sends a request to the SCD containing the HK cryptogram and the second unique ID2 over a secured channel. The SDC decrypts the HK using the MK, generates seed s2 using the HMAC algorithm with the HK and the unique ID2, and sends the seed s2 to server S2 over a secured channel. The server generates DK2 using a KDF with the seed s2 and installs the DK2 into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and the seed s2. When any of the servers are restarted its DK is erased from memory, but the servers can regenerate their DK by reacquiring the seed at any time, by resending the same HK cryptogram and unique ID to the SCD.

### **8.2.5 Multiple HMAC keys with multiple servers**

Multiple servers deployed regionally in different data-centers might need a unique HMAC key per server instead of sharing the same key on all servers. The SCD generates a single MK and multiple HK, one per each server, encrypts each HK using the MK, and sends each HK cryptogram to the corresponding server over a secured channel. The servers store the HK cryptogram and at some previous or subsequent point in time each server generates a unique identifier, ID1 for server S1 and ID2 for server S2. Meanwhile, the SCD destroys the HKs but retains the MK.

To obtain a seed s1 for DK1 the server S1 sends a request to the SCD containing the HK1 cryptogram and the unique ID1 over a secured channel. The SDC decrypts HK1 using the MK, generates seed s1 using the HMAC algorithm with HK1 and the unique ID1, and sends the seed s1 to server S1 over a secured channel. The server S1 generates DK1 using a KDF with the seed s1 and installs the DK1 into memory for data encryption and decryption. Meanwhile, the SCD destroys HK1 and the seed s1.

To obtain a seed s2 for DK2 the server S2 sends a request to the SCD containing the HK2 cryptogram and the second unique ID2 over a secured channel. The SDC decrypts HK2 using the MK, generates seed s2 using the HMAC algorithm with HK2 and the unique ID2, and sends the seed s2 to server S2 over a secured channel. The server generates DK2 using a KDF with the seed s2 and installs the DK2 into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and the seed s2. When any of the servers are restarted its DK is erased from memory, but the servers can regenerate the DK by reacquiring the seed at any time by resending the same HK cryptogram and unique ID to the SCD.

## Annex A

### ASN.1 modules

(This annex forms an integral part of this Recommendation | International Standard.)

#### 1 Main CMS module (From RFC 6268)

```
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) 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
-- 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,
    encapsulatedContentInfo 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, |
      aa-signerInfo | aa-signerInfos | aa-contentLocation ... }

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 ]],
    [[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
        {{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
    [[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 ::= { 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 and 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-signerInfo ATTRIBUTE ::=
    { TYPE SignerInfo IDENTIFIED BY id-signerInfo }
```

```
id-signerInfo OBJECT IDENTIFIER ::= { itu-t recommendation(0) x(24)
  cms-profile(894)attribute(2) signerInfo(0) }

aa-signerInfos ATTRIBUTE ::=
  { TYPE SignerInfos IDENTIFIED BY id-signerInfos }

id-signerInfos OBJECT IDENTIFIER ::= { itu-t recommendation(0) x(24)
  cms-profile(894)attribute(2) signerInfos(1) }

aa-contentLocation ATTRIBUTE ::=
  { TYPE URI IDENTIFIED BY id-contentLocation }

URI ::= UTF8String

id-contentLocation OBJECT IDENTIFIER ::= { itu-t recommendation(0) x(24)
  cms-profile(894)attribute(2) contentLocation(2) }

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
```

## 2 Module AlgorithmInformation-2009 (From RFC 5912)

```
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 is 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 Module CryptographicMessageSyntaxAlgorithms-2009 (From RFC 5911)

```
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 ::= dhpnumber

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 Module PKIX-Algs-2009 (From RFC 5912)

```
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) }

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

IMPORTS

PUBLIC-KEY, SIGNATURE-ALGORITHM, DIGEST-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)}
```

```
mda-sha224, mda-sha256, mda-sha384, mda-sha512
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)} ;
```

```
--
-- Public Key (pk-) Algorithms
--
```

```
PublicKeys PUBLIC-KEY ::= {
    pk-rsa      |
    pk-dsa      |
    pk-dh       |
    pk-kea,
    ...,
    pk-ec       |
    pk-ecDH     |
    pk-ecMQV
}
```

```
--
-- Signature Algorithms (sa-)
--
```

```
SignatureAlgs SIGNATURE-ALGORITHM ::= {
    sa-rsaWithMD2      |
    sa-rsaWithMD5      |
    sa-rsaWithSHA1     |
    sa-dsaWithSHA1     |
    sa-ecdsaWithSHA1,
    ..., -- Extensible
    sa-dsaWithSHA224   |
    sa-dsaWithSHA256   |
    sa-ecdsaWithSHA224 |
    sa-ecdsaWithSHA256 |
    sa-ecdsaWithSHA384 |
    sa-ecdsaWithSHA512
}
```

```
--
-- S/MIME CAPS for algorithms in this document
--
-- For all of the algorithms laid out in this document, the
-- parameters field for the S/MIME capabilities is defined as
-- ABSENT as there are no specific values that need to be known
-- by the receiver for negotiation.
--
```

```
SMimeCaps SMIME-CAPS ::= {
    sa-rsaWithMD2.&smimeCaps |
    sa-rsaWithMD5.&smimeCaps |
    sa-rsaWithSHA1.&smimeCaps |
    sa-dsaWithSHA1.&smimeCaps |
    sa-dsaWithSHA224.&smimeCaps |
    sa-dsaWithSHA256.&smimeCaps |
    sa-ecdsaWithSHA1.&smimeCaps |
    sa-ecdsaWithSHA224.&smimeCaps |
    sa-ecdsaWithSHA256.&smimeCaps |
    sa-ecdsaWithSHA384.&smimeCaps |
    sa-ecdsaWithSHA512.&smimeCaps,
    ... }
```

```
-- RSA PK Algorithm, Parameters, and Keys
```

```
pk-rsa PUBLIC-KEY ::= {
    IDENTIFIER rsaEncryption
```



```
KEY RSAPublicKey
PARAMS TYPE NULL ARE absent
-- Private key format not in this module --
CERT-KEY-USAGE {digitalSignature, nonRepudiation,
    keyEncipherment, dataEncipherment, keyCertSign, cRLSign}
}

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

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

-- DSA PK Algorithm, Parameters, and Keys
pk-dsa PUBLIC-KEY ::= {
    IDENTIFIER id-dsa
    KEY DSAPublicKey
    PARAMS TYPE DSA-Params ARE inheritable
    -- Private key format not in this module --
    CERT-KEY-USAGE { digitalSignature, nonRepudiation, keyCertSign,
        cRLSign }
}

id-dsa OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) x9-57(10040) x9algorithm(4) 1 }

DSA-Params ::= SEQUENCE {
    p  INTEGER,
    q  INTEGER,
    g  INTEGER
}

DSAPublicKey ::= INTEGER -- public key, y
-- Diffie-Hellman PK Algorithm, Parameters, and Keys
pk-dh PUBLIC-KEY ::= {
    IDENTIFIER dhpublicnumber
    KEY DHPublicKey
    PARAMS TYPE DomainParameters ARE inheritable
    -- Private key format not in this module --
    CERT-KEY-USAGE {keyAgreement, encipherOnly, decipherOnly }
}

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

DomainParameters ::= SEQUENCE {
    p          INTEGER,          -- odd prime, p=jq +1
    g          INTEGER,          -- generator, g
    q          INTEGER,          -- factor of p-1
    j          INTEGER OPTIONAL, -- subgroup factor, j>= 2
    validationParams  ValidationParams OPTIONAL
}

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

DHPublicKey ::= INTEGER -- public key, y = g^x mod p
-- KEA PK Algorithm and Parameters

pk-kea PUBLIC-KEY ::= {
    IDENTIFIER id-keyExchangeAlgorithm
    -- key is not encoded --
    PARAMS TYPE KEA-Params-Id ARE required
    -- Private key format not in this module --
    CERT-KEY-USAGE {keyAgreement, encipherOnly, decipherOnly }
}
```

```
id-keyExchangeAlgorithm OBJECT IDENTIFIER ::= {
    joint-iso-itu-t(2) country(16) us(840) organization(1)
    gov(101) dod(2) infosec(1) algorithms(1) 22 }

KEA-Params-Id ::= OCTET STRING

-- Elliptic Curve (EC) Signatures: Unrestricted Algorithms
-- (Section 2.1.1 of RFC 5480)
--
-- EC Unrestricted Algorithm ID -- -- this is used for ECDSA

pk-ec PUBLIC-KEY ::= {
    IDENTIFIER id-ecPublicKey
    KEY ECPoint
    PARAMS TYPE ECParameters ARE required
    -- Private key format not in this module --
    CERT-KEY-USAGE { digitalSignature, nonRepudiation, keyAgreement,
        keyCertSign, cRLSign }
}

ECPoint ::= OCTET STRING -- see RFC 5480 for syntax and restrictions

id-ecPublicKey OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) ansi-X9-62(10045) keyType(2) 1 }

-- Elliptic Curve (EC) Signatures: Restricted Algorithms
-- (Section 2.1.2 of RFC 5480)
--
-- EC Diffie-Hellman Algorithm ID

pk-ecDH PUBLIC-KEY ::= {
    IDENTIFIER id-ecDH
    KEY ECPoint
    PARAMS TYPE ECParameters ARE required
    -- Private key format not in this module --
    CERT-KEY-USAGE { keyAgreement, encipherOnly, decipherOnly }
}

id-ecDH OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) schemes(1) ecdh(12) }

-- EC Menezes-Qu-Vanstone Algorithm ID

pk-ecMQV PUBLIC-KEY ::= {
    IDENTIFIER id-ecMQV
    KEY ECPoint
    PARAMS TYPE ECParameters ARE required
    -- Private key format not in this module --
    CERT-KEY-USAGE { keyAgreement, encipherOnly, decipherOnly }
}

id-ecMQV OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) schemes(1) ecmqv(13) }

-- Parameters and Keys for both Restricted and Unrestricted EC

ECParameters ::= CHOICE {
    namedCurve      CURVE.&id({NamedCurve})
    -- implicitCurve  NULL
    -- implicitCurve MUST NOT be used in PKIX
    -- specifiedCurve SpecifiedCurve
    -- specifiedCurve MUST NOT be used in PKIX
    -- Details for specifiedCurve can be found in [X9.62]
    -- Any future additions to this CHOICE should be coordinated
    -- with ANSI X.9.
}

-- If you need to be able to decode ANSI X.9 parameter structures,
-- uncomment the implicitCurve and specifiedCurve above, and also
-- uncomment the following:
```

```
--(WITH COMPONENTS {namedCurve PRESENT})
-- Sec 2.1.1.1 Named Curve
CURVE ::= CLASS { &id OBJECT IDENTIFIER UNIQUE }
    WITH SYNTAX { ID &id }

NamedCurve CURVE ::= {
    { ID secp192r1 } | { ID sect163k1 } | { ID sect163r2 } |
    { ID secp224r1 } | { ID sect233k1 } | { ID sect233r1 } |
    { ID secp256r1 } | { ID sect283k1 } | { ID sect283r1 } |
    { ID secp384r1 } | { ID sect409k1 } | { ID sect409r1 } |
    { ID secp521r1 } | { ID sect571k1 } | { ID sect571r1 },
    ... -- Extensible
}

-- Note in [X9.62] the curves are referred to as 'ansiX9' as
-- opposed to 'sec'. For example, secp192r1 is the same curve as
-- ansix9p192r1.
-- Note that in [PKI-ALG] the secp192r1 curve was referred to as
-- prime192v1 and the secp256r1 curve was referred to as
-- prime256v1.
-- Note that [FIPS186-3] refers to secp192r1 as P-192,
-- secp224r1 as P-224, secp256r1 as P-256, secp384r1 as P-384,
-- and secp521r1 as P-521.

secp192r1 OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) ansi-X9-62(10045) curves(3)
    prime(1) 1 }

sect163k1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 1 }

sect163r2 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 15 }

secp224r1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 33 }

sect233k1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 26 }

sect233r1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 27 }

secp256r1 OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) ansi-X9-62(10045) curves(3)
    prime(1) 7 }

sect283k1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 16 }

sect283r1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 17 }

secp384r1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 34 }

sect409k1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 36 }

sect409r1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 37 }

secp521r1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 35 }

sect571k1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 38 }

sect571r1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) certicom(132) curve(0) 39 }

-- RSA with MD-2
sa-rsaWithMD2 SIGNATURE-ALGORITHM ::= {
```

```
IDENTIFIER md2WithRSAEncryption
PARAMS TYPE NULL ARE required
HASHES { mda-md2 }
PUBLIC-KEYS { pk-rsa }
SMIME-CAPS { IDENTIFIED BY md2WithRSAEncryption }
}

md2WithRSAEncryption OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
    pkcs-1(1) 2 }

-- RSA with MD-5

sa-rsaWithMD5 SIGNATURE-ALGORITHM ::= {
    IDENTIFIER md5WithRSAEncryption
    PARAMS TYPE NULL ARE required
    HASHES { mda-md5 }
    PUBLIC-KEYS { pk-rsa }
    SMIME-CAPS { IDENTIFIED BY md5WithRSAEncryption }
}

md5WithRSAEncryption OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 4 }

-- RSA with SHA-1

sa-rsaWithSHA1 SIGNATURE-ALGORITHM ::= {
    IDENTIFIER sha1WithRSAEncryption
    PARAMS TYPE NULL ARE required
    HASHES { mda-sha1 }
    PUBLIC-KEYS { pk-rsa }
    SMIME-CAPS { IDENTIFIED BY sha1WithRSAEncryption }
}

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

-- DSA with SHA-1

sa-dsaWithSHA1 SIGNATURE-ALGORITHM ::= {
    IDENTIFIER dsa-with-sha1
    VALUE DSA-Sig-Value
    PARAMS TYPE NULL ARE absent
    HASHES { mda-sha1 }
    PUBLIC-KEYS { pk-dsa }
    SMIME-CAPS { IDENTIFIED BY dsa-with-sha1 }
}

dsa-with-sha1 OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) x9-57(10040) x9algorithm(4) 3 }

-- DSA with SHA-224

sa-dsaWithSHA224 SIGNATURE-ALGORITHM ::= {
    IDENTIFIER dsa-with-sha224
    VALUE DSA-Sig-Value
    PARAMS TYPE NULL ARE absent
    HASHES { mda-sha224 }
    PUBLIC-KEYS { pk-dsa }
    SMIME-CAPS { IDENTIFIED BY dsa-with-sha224 }
}

dsa-with-sha224 OBJECT IDENTIFIER ::= {
    joint-iso-ccitt(2) country(16) us(840) organization(1) gov(101)
    csor(3) algorithms(4) id-dsa-with-sha2(3) 1 }

-- DSA with SHA-256

sa-dsaWithSHA256 SIGNATURE-ALGORITHM ::= {
    IDENTIFIER dsa-with-sha256
    VALUE DSA-Sig-Value
    PARAMS TYPE NULL ARE absent
    HASHES { mda-sha256 }
    PUBLIC-KEYS { pk-dsa }
```

```
SMIME-CAPS { IDENTIFIED BY dsa-with-sha256 }
}

dsa-with-sha256 OBJECT IDENTIFIER ::= {
    joint-iso-ccitt(2) country(16) us(840) organization(1) gov(101)
    csor(3) algorithms(4) id-dsa-with-sha2(3) 2 }

-- ECDSA with SHA-1

sa-ecdsaWithSHA1 SIGNATURE-ALGORITHM ::= {
    IDENTIFIER ecdsa-with-SHA1
    VALUE ECDSA-Sig-Value
    PARAMS TYPE NULL ARE absent
    HASHES { mda-sha1 }
    PUBLIC-KEYS { pk-ec }
    SMIME-CAPS { IDENTIFIED BY ecdsa-with-SHA1 }
}

ecdsa-with-SHA1 OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) ansi-X9-62(10045)
    signatures(4) 1 }

-- ECDSA with SHA-224

sa-ecdsaWithSHA224 SIGNATURE-ALGORITHM ::= {
    IDENTIFIER ecdsa-with-SHA224
    VALUE ECDSA-Sig-Value
    PARAMS TYPE NULL ARE absent
    HASHES { mda-sha224 }
    PUBLIC-KEYS { pk-ec }
    SMIME-CAPS { IDENTIFIED BY ecdsa-with-SHA224 }
}

ecdsa-with-SHA224 OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4)
    ecdsa-with-SHA2(3) 1 }
    -- ECDSA with SHA-256
    sa-ecdsaWithSHA256 SIGNATURE-ALGORITHM ::= {
        IDENTIFIER ecdsa-with-SHA256
        VALUE ECDSA-Sig-Value
        PARAMS TYPE NULL ARE absent
        HASHES { mda-sha256 }
        PUBLIC-KEYS { pk-ec }
        SMIME-CAPS { IDENTIFIED BY ecdsa-with-SHA256 }
    }

ecdsa-with-SHA256 OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4)
    ecdsa-with-SHA2(3) 2 }

-- ECDSA with SHA-384

sa-ecdsaWithSHA384 SIGNATURE-ALGORITHM ::= {
    IDENTIFIER ecdsa-with-SHA384
    VALUE ECDSA-Sig-Value
    PARAMS TYPE NULL ARE absent
    HASHES { mda-sha384 }
    PUBLIC-KEYS { pk-ec }
    SMIME-CAPS { IDENTIFIED BY ecdsa-with-SHA384 }
}

ecdsa-with-SHA384 OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4)
    ecdsa-with-SHA2(3) 3 }

-- ECDSA with SHA-512

sa-ecdsaWithSHA512 SIGNATURE-ALGORITHM ::= {
    IDENTIFIER ecdsa-with-SHA512
    VALUE ECDSA-Sig-Value
    PARAMS TYPE NULL ARE absent
    HASHES { mda-sha512 }
```

```
PUBLIC-KEYS { pk-ec }
SMIME-CAPS { IDENTIFIED BY ecdsa-with-SHA512 }
}

ecdsa-with-SHA512 OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) ansi-X9-62(10045) signatures(4)
    ecdsa-with-SHA2(3) 4 }

--
-- Signature Values
--
-- DSA

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

-- ECDSA

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

--
-- Message Digest Algorithms (mda-)
--

HashAlgs DIGEST-ALGORITHM ::= {
    mda-md2      |
    mda-md5      |
    mda-sha1,
    ... -- Extensible
}

-- MD-2

mda-md2 DIGEST-ALGORITHM ::= {
    IDENTIFIER id-md2
    PARAMS TYPE NULL ARE preferredAbsent
}

id-md2 OBJECT IDENTIFIER ::= {
    iso(1) member-body(2) us(840) rsadsi(113549)
    digestAlgorithm(2) 2 }

-- MD-5

mda-md5 DIGEST-ALGORITHM ::= {
    IDENTIFIER id-md5
    PARAMS TYPE NULL ARE preferredAbsent
}

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

-- SHA-1

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

id-sha1 OBJECT IDENTIFIER ::= {
    iso(1) identified-organization(3) oiw(14) secsig(3)
    algorithm(2) 26 }

END
```

## 5 Module PKIXAttributeCertificate-2009 (From RFC 5912)

```
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)}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS

AttributeSet{}, Extensions{}, SecurityCategory{}, EXTENSION, ATTRIBUTE,
SECURITY-CATEGORY
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{}, SIGNATURE-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)}

-- IMPORTed module OIDs MAY change if [PKIXPROF] changes
-- PKIX Certificate Extensions

CertificateSerialNumber, UniqueIdentifier, id-pkix, id-pe, id-kp,
    id-ad, id-at, SIGNED{}, SignatureAlgorithms
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)}

GeneralName, GeneralNames, id-ce, ext-AuthorityKeyIdentifier,
    ext-AuthorityInfoAccess, ext-CRLDistributionPoints
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)}

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) };

-- Define the set of extensions that can appear.
-- Some of these are imported from PKIX Cert

AttributeCertExtensions EXTENSION ::= {
    ext-auditIdentity          |
    ext-targetInformation      |
    ext-AuthorityKeyIdentifier |
    ext-AuthorityInfoAccess    |
    ext-CRLDistributionPoints  |
    ext-noRevAvail            |
    ext-ac-proxying            |
    ext-aaControls,           |
    ... }

ext-auditIdentity EXTENSION ::= {
    SYNTAX OCTET STRING
    IDENTIFIED BY id-pe-ac-auditIdentity}

ext-targetInformation EXTENSION ::= {
    SYNTAX Targets
    IDENTIFIED BY id-ce-targetInformation }

ext-noRevAvail EXTENSION ::= {
    SYNTAX NULL
    IDENTIFIED BY id-ce-noRevAvail}

ext-ac-proxying EXTENSION ::= {
    SYNTAX ProxyInfo
    IDENTIFIED BY id-pe-ac-proxying}
```

```
ext-aaControls EXTENSION ::= {
    SYNTAX AAControls
    IDENTIFIED BY id-pe-aaControls}

-- Define the set of attributes used here

AttributesDefined ATTRIBUTE ::= { at-authenticationInfo |
    at-accesIdentity          |
    at-chargingIdentity       |
    at-group                  |
    at-role                   |
    at-clearance              |
    at-encAttrs, ...}

at-authenticationInfo ATTRIBUTE ::= {
    TYPE SvceAuthInfo
    IDENTIFIED BY id-aca-authenticationInfo}

at-accesIdentity ATTRIBUTE ::= {
    TYPE SvceAuthInfo
    IDENTIFIED BY id-aca-accessIdentity}

at-chargingIdentity ATTRIBUTE ::= {
    TYPE IetfAttrSyntax
    IDENTIFIED BY id-aca-chargingIdentity}

at-group ATTRIBUTE ::= {
    TYPE IetfAttrSyntax
    IDENTIFIED BY id-aca-group}

at-role ATTRIBUTE ::= {
    TYPE RoleSyntax
    IDENTIFIED BY id-at-role}

at-clearance ATTRIBUTE ::= {
    TYPE Clearance
    IDENTIFIED BY id-at-clearance}

at-clearance-RFC3281 ATTRIBUTE ::= {
    TYPE Clearance-rfc3281
    IDENTIFIED BY id-at-clearance-rfc3281 }

at-encAttrs ATTRIBUTE ::= {
    TYPE ContentInfo
    IDENTIFIED BY id-aca-encAttrs}

--
--  OIDs used by Attribute Certificate Extensions
--
id-pe-ac-auditIdentity      OBJECT IDENTIFIER ::= { id-pe 4 }
id-pe-aaControls            OBJECT IDENTIFIER ::= { id-pe 6 }
id-pe-ac-proxying          OBJECT IDENTIFIER ::= { id-pe 10 }
id-ce-targetInformation     OBJECT IDENTIFIER ::= { id-ce 55 }
id-ce-noRevAvail           OBJECT IDENTIFIER ::= { id-ce 56 }

--
--  OIDs used by Attribute Certificate Attributes
--
id-aca                      OBJECT IDENTIFIER ::= { id-pkix 10 }
id-aca-authenticationInfo   OBJECT IDENTIFIER ::= { id-aca 1 }
id-aca-accessIdentity       OBJECT IDENTIFIER ::= { id-aca 2 }
id-aca-chargingIdentity     OBJECT IDENTIFIER ::= { id-aca 3 }
id-aca-group                OBJECT IDENTIFIER ::= { id-aca 4 }
-- { id-aca 5 } is reserved
```



```
id-aca-encAttrs          OBJECT IDENTIFIER ::= { id-aca 6 }
id-at-role               OBJECT IDENTIFIER ::= { id-at 72}
id-at-clearance         OBJECT IDENTIFIER ::= {
    joint-iso-ccitt(2) ds(5) attributeType(4) clearance (55) }
-- Uncomment the following declaration and comment the above line if
-- using the id-at-clearance attribute as defined in [RFC3281]
-- id-at-clearance ::= id-at-clearance-3281
id-at-clearance-rfc3281 OBJECT IDENTIFIER ::= {
    joint-iso-itu-t(2) ds(5) module(1) selected-attribute-types(5)
    clearance (55) }
--
-- The syntax of an Attribute Certificate
--
AttributeCertificate ::= SIGNED{AttributeCertificateInfo}
AttributeCertificateInfo ::= SEQUENCE {
    version                AttCertVersion, -- version is v2
    holder                 Holder,
    issuer                 AttCertIssuer,
    signature              AlgorithmIdentifier{SIGNATURE-ALGORITHM,
        {SignatureAlgorithms}},
    serialNumber           CertificateSerialNumber,
    attrCertValidityPeriod AttCertValidityPeriod,
    attributes             SEQUENCE OF
        AttributeSet{{AttributesDefined}},
    issuerUniqueID         UniqueIdentifier OPTIONAL,
    extensions             Extensions{{AttributeCertExtensions}} OPTIONAL
}
AttCertVersion ::= INTEGER { v2(1) }
Holder ::= SEQUENCE {
    baseCertificateID      [0] IssuerSerial OPTIONAL,
    -- the issuer and serial number of
    -- the holder's Public Key Certificate
    entityName             [1] GeneralNames OPTIONAL,
    -- the name of the claimant or role
    objectDigestInfo       [2] ObjectDigestInfo OPTIONAL
    -- used to directly authenticate the
    -- holder, for example, an executable
}
ObjectDigestInfo ::= SEQUENCE {
    digestedObjectType     ENUMERATED {
        publicKey          (0),
        publicKeyCert      (1),
        otherObjectTypes   (2) },
    -- otherObjectTypes MUST NOT
    -- be used in this profile
    otherObjectTypeID     OBJECT IDENTIFIER OPTIONAL,
    digestAlgorithm        AlgorithmIdentifier{DIGEST-ALGORITHM, {...}},
    objectDigest           BIT STRING
}
AttCertIssuer ::= CHOICE {
    v1Form                 GeneralNames, -- MUST NOT be used in this
    -- profile
    v2Form                 [0] V2Form    -- v2 only
}
V2Form ::= SEQUENCE {
    issuerName             GeneralNames OPTIONAL,
    baseCertificateID      [0] IssuerSerial OPTIONAL,
    objectDigestInfo       [1] ObjectDigestInfo OPTIONAL
    -- issuerName MUST be present in this profile
}
```

```
-- baseCertificateID and objectDigestInfo MUST
-- NOT be present in this profile
}

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

AttCertValidityPeriod ::= SEQUENCE {
    notBeforeTime GeneralizedTime,
    notAfterTime  GeneralizedTime
}

--
-- Syntax used by Attribute Certificate Extensions
--

Targets ::= SEQUENCE OF Target

Target ::= CHOICE {
    targetName  [0] GeneralName,
    targetGroup [1] GeneralName,
    targetCert  [2] TargetCert
}

TargetCert ::= SEQUENCE {
    targetCertificate IssuerSerial,
    targetName        GeneralName OPTIONAL,
    certDigestInfo   ObjectDigestInfo OPTIONAL
}

AACControls ::= SEQUENCE {
    pathLenConstraint INTEGER (0..MAX) OPTIONAL,
    permittedAttrs    [0] AttrSpec OPTIONAL,
    excludedAttrs     [1] AttrSpec OPTIONAL,
    permitUnspecified BOOLEAN DEFAULT TRUE
}

AttrSpec ::= SEQUENCE OF OBJECT IDENTIFIER

ProxyInfo ::= SEQUENCE OF Targets

--
-- Syntax used by Attribute Certificate Attributes
--

IetfAttrSyntax ::= SEQUENCE {
    policyAuthority [0] GeneralNames OPTIONAL,
    values          SEQUENCE OF CHOICE {
        octets  OCTET STRING,
        oid     OBJECT IDENTIFIER,
        string  UTF8String
    }
}

SvcAuthInfo ::= SEQUENCE {
    service  GeneralName,
    ident    GeneralName,
    authInfo OCTET STRING OPTIONAL
}

RoleSyntax ::= SEQUENCE {
    roleAuthority [0] GeneralNames OPTIONAL,
    roleName     [1] GeneralName
}

Clearance ::= SEQUENCE {
    policyId          OBJECT IDENTIFIER,
    classList         ClassList DEFAULT {unclassified},
    securityCategories SET OF SecurityCategory
    {{SupportedSecurityCategories}} OPTIONAL
}
```

```
    }

-- Uncomment the following lines to support deprecated clearance
-- syntax and comment out previous Clearance.
-- Clearance ::= Clearance-rfc3281

Clearance-rfc3281 ::= SEQUENCE {
    policyId          [0] OBJECT IDENTIFIER,
    classList         [1] ClassList DEFAULT {unclassified},
    securityCategories [2] SET OF SecurityCategory-rfc3281
        {{SupportedSecurityCategories}} OPTIONAL
    }

ClassList ::= BIT STRING {
    unmarked          (0),
    unclassified      (1),
    restricted        (2),
    confidential      (3),
    secret            (4),
    topSecret         (5)
    }

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

SecurityCategory-rfc3281{SECURITY-CATEGORY:Supported} ::= SEQUENCE {
    type             [0] IMPLICIT SECURITY-CATEGORY.&id({Supported}),
    value           [1] EXPLICIT SECURITY-CATEGORY.&Type({Supported}{@type})
    }

ACClearAttrs ::= SEQUENCE {
    acIssuer         GeneralName,
    acSerial         INTEGER,
    attrs           SEQUENCE OF AttributeSet{{AttributesDefined}}
    }

END
```

## 6 Module AttributeCertificateVersion1-2009 (From RFC 5912)

```
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)}

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

IMPORTS

SIGNATURE-ALGORITHM, 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)}

AttributeSet{ }, Extensions{ }, EXTENSION, 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) }

CertificateSerialNumber, UniqueIdentifier, SIGNED{
    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) }

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) }

AttCertValidityPeriod, IssuerSerial
    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) } ;
```

```
-- Definition extracted from X.509-1997 [X.509-97], but
-- different type names are used to avoid collisions.
```

```
AttributeCertificateV1 ::= SIGNED{AttributeCertificateInfoV1}
AttributeCertificateInfoV1 ::= SEQUENCE {
    version      AttCertVersionV1 DEFAULT v1,
    subject      CHOICE {
        baseCertificateID      [0] IssuerSerial,
        -- associated with a Public Key Certificate
        subjectName             [1] GeneralNames },
    -- associated with a name
    issuer          GeneralNames,
    signature       AlgorithmIdentifier{SIGNATURE-ALGORITHM, {...}},
    serialNumber    CertificateSerialNumber,
    attCertValidityPeriod AttCertValidityPeriod,
    attributes      SEQUENCE OF AttributeSet{{AttrList}},
    issuerUniqueID  UniqueIdentifier OPTIONAL,
    extensions      Extensions{{AttributeCertExtensionsV1}} OPTIONAL
}

AttCertVersionV1 ::= INTEGER { v1(0) }

AttrList ATTRIBUTE ::= {...}

AttributeCertExtensionsV1 EXTENSION ::= {...}

END
```

## 7 Module PKIX-CommonTypes-2009 (From RFC 5912)

```
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)}
```

```
DEFINITIONS EXPLICIT TAGS ::=
```

```
BEGIN
```

```
-- ATTRIBUTE
--
-- Describe the set of data associated with an attribute of some type
--
-- &id is an OID identifying the attribute
-- &Type is the ASN.1 type structure for the attribute; not all
-- attributes have a data structure, so this field is optional
-- &minCount contains the minimum number of times the attribute can
-- occur in an AttributeSet
-- &maxCount contains the maximum number of times the attribute can
-- appear in an AttributeSet
-- Note: this cannot be automatically enforced as the field
-- cannot be defaulted to MAX.
-- &equality-match contains information about how matching should be
-- done
--
-- Currently we are using two different prefixes for attributes.
--
-- at- for certificate attributes
-- aa- for CMS attributes
--
```

```
ATTRIBUTE ::= CLASS {
    &id          OBJECT IDENTIFIER UNIQUE,
    &Type        OPTIONAL,
    &equality-match MATCHING-RULE OPTIONAL,
    &minCount    INTEGER DEFAULT 1,
    &maxCount    INTEGER OPTIONAL
}
```

```
    } WITH SYNTAX {  
      [TYPE &Type]  
      [EQUALITY MATCHING RULE &equality-match]  
      [COUNTS [MIN &minCount] [MAX &maxCount]]  
      IDENTIFIED BY &id  
    }
```

-- Specification of MATCHING-RULE information object class  
--

```
MATCHING-RULE ::= CLASS {  
    &ParentMatchingRules      MATCHING-RULE OPTIONAL,  
    &AssertionType            OPTIONAL,  
    &uniqueMatchIndicator     ATTRIBUTE OPTIONAL,  
    &id                       OBJECT IDENTIFIER UNIQUE  
}  
WITH SYNTAX {  
    [PARENT &ParentMatchingRules]  
    [SYNTAX &AssertionType]  
    [UNIQUE-MATCH-INDICATOR &uniqueMatchIndicator]  
    ID &id  
}
```

-- AttributeSet

--

-- Used when a set of attributes is to occur.

--

-- type contains the identifier of the attribute

-- values contains a set of values where the structure of the ASN.1

-- is defined by the attribute

--

-- The parameter contains the set of objects describing

-- those attributes that can occur in this location.

--

```
AttributeSet{ATTRIBUTE:AttrSet} ::= SEQUENCE {  
    type          ATTRIBUTE.&id({AttrSet}),  
    values        SET SIZE (1..MAX) OF ATTRIBUTE.&Type({AttrSet}{@type})  
}
```

-- SingleAttribute

--

-- Used for a single valued attribute

--

-- The parameter contains the set of objects describing the

-- attributes that can occur in this location

--

```
SingleAttribute{ATTRIBUTE:AttrSet} ::= SEQUENCE {  
    type          ATTRIBUTE.&id({AttrSet}),  
    value         ATTRIBUTE.&Type({AttrSet}{@type})  
}
```

-- EXTENSION

--

-- This class definition is used to describe the association of  
-- object identifier and ASN.1 type structure for extensions

--

-- All extensions are prefixed with ext-

--

-- &id contains the object identifier for the extension

-- &ExtnType specifies the ASN.1 type structure for the extension

-- &Critical contains the set of legal values for the critical field.

-- This is normally {TRUE|FALSE} but in some instances may be  
-- restricted to just one of these values.

--

```
EXTENSION ::= CLASS {  
    &id          OBJECT IDENTIFIER UNIQUE,  
    &ExtnType,  
    &Critical    BOOLEAN DEFAULT {TRUE | FALSE }
```

```
    } WITH SYNTAX {
      SYNTAX &ExtnType IDENTIFIED BY &id
      [CRITICALITY &Critical]
    }

-- Extensions
--
-- Used for a sequence of extensions.
--
-- The parameter contains the set of legal extensions that can
-- occur in this sequence.
--
Extensions{EXTENSION:ExtensionSet} ::=
    SEQUENCE SIZE (1..MAX) OF Extension{{ExtensionSet}}

-- Extension
--
-- Used for a single extension
--
-- The parameter contains the set of legal extensions that can
-- occur in this extension.
--
-- The restriction on the critical field has been commented out
-- the authors are not completely sure it is correct.
-- The restriction could be done using custom code rather than
-- compiler-generated code, however.
--
Extension{EXTENSION:ExtensionSet} ::= SEQUENCE {
    extnID      EXTENSION.&id({ExtensionSet}),
    critical    BOOLEAN
    --          (EXTENSION.&Critical({ExtensionSet}{@extnID}))
    --          DEFAULT FALSE,
    extnValue   OCTET STRING (CONTAINING
    --          EXTENSION.&ExtnType({ExtensionSet}{@extnID}))
    -- contains the DER encoding of the ASN.1 value
    -- corresponding to the extension type identified
    -- by extnID
}

-- Security Category
--
-- Security categories are used both for specifying clearances and
-- for labeling objects. We move this here from RFC 3281 so that
-- they will use a common single object class to express this
-- information.
--
SECURITY-CATEGORY ::= TYPE-IDENTIFIER

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

END
```

## 8 Module PKIX-X400Address-2009 (From RFC 5912)

```
PKIX-X400Address-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-x400address-02(60) }

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- X.400 address syntax starts here
```

```
ORAddress ::= SEQUENCE {
    built-in-standard-attributes    BuiltInStandardAttributes,
    built-in-domain-defined-attributes BuiltInDomainDefinedAttributes
    OPTIONAL,
    -- see also teletex-domain-defined-attributes
    extension-attributes            ExtensionAttributes OPTIONAL
}

-- Built-in Standard Attributes

BuiltInStandardAttributes ::= SEQUENCE {
    country-name                    CountryName OPTIONAL,
    administration-domain-name      AdministrationDomainName OPTIONAL,
    network-address                  [0] IMPLICIT NetworkAddress OPTIONAL,
    -- see also extended-network-address
    terminal-identifier               [1] IMPLICIT TerminalIdentifier OPTIONAL,
    private-domain-name              [2] PrivateDomainName OPTIONAL,
    organization-name                [3] IMPLICIT OrganizationName OPTIONAL,
    -- see also teletex-organization-name
    numeric-user-identifier           [4] IMPLICIT NumericUserIdentifier
    OPTIONAL,
    personal-name                    [5] IMPLICIT PersonalName OPTIONAL,
    -- see also teletex-personal-name
    organizational-unit-names        [6] IMPLICIT OrganizationalUnitNames
    OPTIONAL
}

-- see also teletex-organizational-unit-names

CountryName ::= [APPLICATION 1] CHOICE {
    x121-dcc-code                    NumericString
        (SIZE (ub-country-name-numeric-length)),
    iso-3166-alpha2-code             PrintableString
        (SIZE (ub-country-name-alpha-length))
}

AdministrationDomainName ::= [APPLICATION 2] CHOICE {
    numeric    NumericString(SIZE (0..ub-domain-name-length)),
    printable  PrintableString(SIZE (0..ub-domain-name-length))
}

NetworkAddress ::= X121Address -- see also extended-network-address

X121Address ::= NumericString (SIZE (1..ub-x121-address-length))

TerminalIdentifier ::= PrintableString (SIZE(1..ub-terminal-id-length))

PrivateDomainName ::= CHOICE {
    numeric    NumericString(SIZE (1..ub-domain-name-length)),
    printable  PrintableString(SIZE (1..ub-domain-name-length))
}

OrganizationName ::= PrintableString(SIZE (1..ub-organization-name-length))

-- see also teletex-organization-name

NumericUserIdentifier ::= NumericString(SIZE (1..ub-numeric-user-id-length))

PersonalName ::= SET {
    surname                    [0] IMPLICIT PrintableString
        (SIZE (1..ub-surname-length)),
    given-name                  [1] IMPLICIT PrintableString
        (SIZE (1..ub-given-name-length)) OPTIONAL,
    initials                    [2] IMPLICIT PrintableString
        (SIZE (1..ub-initials-length)) OPTIONAL,
    generation-qualifier        [3] IMPLICIT PrintableString
        (SIZE (1..ub-generation-qualifier-length)) OPTIONAL
}

-- see also teletex-personal-name

OrganizationalUnitNames ::= SEQUENCE SIZE (1..ub-organizational-units)
    OF OrganizationalUnitName
```

```
-- see also teletex-organizational-unit-names
OrganizationalUnitName ::= PrintableString
    (SIZE(1..ub-organizational-unit-name-length))

-- Built-in Domain-defined Attributes
BuiltInDomainDefinedAttributes ::= SEQUENCE SIZE
    (1..ub-domain-defined-attributes) OF BuiltInDomainDefinedAttribute

BuiltInDomainDefinedAttribute ::= SEQUENCE {
    type PrintableString
    (SIZE(1..ub-domain-defined-attribute-type-length)),
    value PrintableString
    (SIZE(1..ub-domain-defined-attribute-value-length))
}

-- Extension Attributes
ExtensionAttributes ::= SET SIZE (1..ub-extension-attributes) OF
    ExtensionAttribute

EXTENSION-ATTRIBUTE ::= CLASS {
    &id INTEGER (0..ub-extension-attributes) UNIQUE,
    &Type
} WITH SYNTAX { &Type IDENTIFIED BY &id }

ExtensionAttribute ::= SEQUENCE {
    extension-attribute-type [0] IMPLICIT EXTENSION-ATTRIBUTE.
        &id({SupportedExtensionAttributes}),
    extension-attribute-value [1] EXTENSION-ATTRIBUTE.
        &Type({SupportedExtensionAttributes}{@extension-attribute-type})
}

SupportedExtensionAttributes EXTENSION-ATTRIBUTE ::= {
    ea-commonName | ea-teletexCommonName | ea-teletexOrganizationName |
    ea-teletexPersonalName | ea-teletexOrganizationalUnitNames |
    ea-pDSName | ea-physicalDeliveryCountryName | ea-postalCode |
    ea-physicalDeliveryOfficeName | ea-physicalDeliveryOfficeNumber |
    ea-extensionORAddressComponents | ea-physicalDeliveryPersonalName |
    ea-physicalDeliveryOrganizationName |
    ea-extensionPhysicalDeliveryAddressComponents |
    ea-unformattedPostalAddress | ea-streetAddress |
    ea-postOfficeBoxAddress | ea-posteRestanteAddress |
    ea-uniquePostalName | ea-localPostalAttributes |
    ea-extendedNetworkAddress | ea-terminalType |
    ea-teletexDomainDefinedAttributes,
    ...
}

-- Extension types and attribute values
ea-commonName EXTENSION-ATTRIBUTE ::= {
    PrintableString (SIZE (1..ub-common-name-length)) IDENTIFIED BY 1 }

ea-teletexCommonName EXTENSION-ATTRIBUTE ::= {
    TeletexString(SIZE (1..ub-common-name-length)) IDENTIFIED BY 2 }

ea-teletexOrganizationName EXTENSION-ATTRIBUTE ::= {
    TeletexString(SIZE (1..ub-organization-name-length)) IDENTIFIED BY 3 }

ea-teletexPersonalName EXTENSION-ATTRIBUTE ::= {
    SET {
        surname [0] IMPLICIT TeletexString
            (SIZE (1..ub-surname-length)),
        given-name [1] IMPLICIT TeletexString
            (SIZE (1..ub-given-name-length)) OPTIONAL,
        initials [2] IMPLICIT TeletexString
            (SIZE (1..ub-initials-length)) OPTIONAL,
        generation-qualifier [3] IMPLICIT TeletexString
            (SIZE (1..ub-generation-qualifier-length)) OPTIONAL
    }
    IDENTIFIED BY 4 }
```



```
ea-teletexOrganizationalUnitNames EXTENSION-ATTRIBUTE ::= {
    SEQUENCE SIZE (1..ub-organizational-units) OF
        TeletexOrganizationalUnitName
    IDENTIFIED BY 5 }

TeletexOrganizationalUnitName ::= TeletexString
    (SIZE (1..ub-organizational-unit-name-length))

ea-pdsName EXTENSION-ATTRIBUTE ::= {PrintableString
    (SIZE (1..ub-pds-name-length)) IDENTIFIED BY 7 }

ea-physicalDeliveryCountryName EXTENSION-ATTRIBUTE ::= {
    CHOICE {
        x121-dcc-code      NumericString
            (SIZE (ub-country-name-numeric-length)),
        iso-3166-alpha2-code PrintableString
            (SIZE (ub-country-name-alpha-length))
    }
    IDENTIFIED BY 8 }

ea-postalCode EXTENSION-ATTRIBUTE ::= {
    CHOICE {
        numeric-code      NumericString (SIZE (1..ub-postal-code-length)),
        printable-code    PrintableString (SIZE (1..ub-postal-code-length))
    }
    IDENTIFIED BY 9 }

ea-physicalDeliveryOfficeName EXTENSION-ATTRIBUTE ::=
    { PDSParameter IDENTIFIED BY 10 }

ea-physicalDeliveryOfficeNumber EXTENSION-ATTRIBUTE ::=
    {PDSParameter IDENTIFIED BY 11 }

ea-extensionORAddressComponents EXTENSION-ATTRIBUTE ::=
    {PDSParameter IDENTIFIED BY 12 }

ea-physicalDeliveryPersonalName EXTENSION-ATTRIBUTE ::=
    {PDSParameter IDENTIFIED BY 13}

ea-physicalDeliveryOrganizationName EXTENSION-ATTRIBUTE ::=
    {PDSParameter IDENTIFIED BY 14 }

ea-extensionPhysicalDeliveryAddressComponents EXTENSION-ATTRIBUTE ::=
    {PDSParameter IDENTIFIED BY 15 }

ea-unformattedPostalAddress EXTENSION-ATTRIBUTE ::= {
    SET {
        printable-address      SEQUENCE SIZE (1..ub-pds-physical-address-lines)
            OF PrintableString (SIZE (1..ub-pds-parameter-length)) OPTIONAL,
        teletex-string         TeletexString (SIZE (1..ub-unformatted-address-length))
            OPTIONAL
    }
    IDENTIFIED BY 16 }

ea-streetAddress EXTENSION-ATTRIBUTE ::=
    {PDSParameter IDENTIFIED BY 17 }

ea-postOfficeBoxAddress EXTENSION-ATTRIBUTE ::=
    {PDSParameter IDENTIFIED BY 18 }

ea-posteRestanteAddress EXTENSION-ATTRIBUTE ::=
    {PDSParameter IDENTIFIED BY 19 }

ea-uniquePostalName EXTENSION-ATTRIBUTE ::=
    { PDSParameter IDENTIFIED BY 20 }

ea-localPostalAttributes EXTENSION-ATTRIBUTE ::=
    {PDSParameter IDENTIFIED BY 21 }

PDSParameter ::= SET {
    printable-string      PrintableString
        (SIZE(1..ub-pds-parameter-length)) OPTIONAL,
    teletex-string        TeletexString
        (SIZE(1..ub-pds-parameter-length)) OPTIONAL
```

```
    }  
ea-extendedNetworkAddress EXTENSION-ATTRIBUTE ::= {  
    CHOICE {  
        e163-4-address SEQUENCE {  
            number      [0] IMPLICIT NumericString  
                (SIZE (1..ub-e163-4-number-length)),  
            sub-address [1] IMPLICIT NumericString  
                (SIZE (1..ub-e163-4-sub-address-length)) OPTIONAL  
        },  
        psap-address   [0] IMPLICIT PresentationAddress  
    }  
    IDENTIFIED BY 22  
}  
  
PresentationAddress ::= SEQUENCE {  
    pSelector      [0] EXPLICIT OCTET STRING OPTIONAL,  
    sSelector      [1] EXPLICIT OCTET STRING OPTIONAL,  
    tSelector      [2] EXPLICIT OCTET STRING OPTIONAL,  
    nAddresses     [3] EXPLICIT SET SIZE (1..MAX) OF OCTET STRING  
}  
  
ea-terminalType    EXTENSION-ATTRIBUTE ::= {INTEGER {  
    telex (3),  
    teletex (4),  
    g3-facsimile (5),  
    g4-facsimile (6),  
    ia5-terminal (7),  
    videotex (8)  
    } (0..ub-integer-options)  
    IDENTIFIED BY 23 }  
  
-- Extension Domain-defined Attributes  
ea-teletexDomainDefinedAttributes EXTENSION-ATTRIBUTE ::= {  
    SEQUENCE SIZE (1..ub-domain-defined-attributes) OF  
        TeletexDomainDefinedAttribute  
    IDENTIFIED BY 6 }  
  
TeletexDomainDefinedAttribute ::= SEQUENCE {  
    type TeletexString (SIZE (1..ub-domain-defined-attribute-type-length)),  
    value TeletexString (SIZE (1..ub-domain-defined-attribute-value-length)) }  
  
-- specifications of Upper Bounds MUST be regarded as mandatory  
-- from Annex B of ITU-T X.411 Reference Definition of MTS Parameter  
-- Upper Bounds  
  
ub-match INTEGER ::= 128  
ub-common-name-length INTEGER ::= 64  
ub-country-name-alpha-length INTEGER ::= 2  
ub-country-name-numeric-length INTEGER ::= 3  
ub-domain-defined-attributes INTEGER ::= 4  
ub-domain-defined-attribute-type-length INTEGER ::= 8  
ub-domain-defined-attribute-value-length INTEGER ::= 128  
ub-domain-name-length INTEGER ::= 16  
ub-extension-attributes INTEGER ::= 256  
ub-e163-4-number-length INTEGER ::= 15  
ub-e163-4-sub-address-length INTEGER ::= 40  
ub-generation-qualifier-length INTEGER ::= 3  
ub-given-name-length INTEGER ::= 16  
ub-initials-length INTEGER ::= 5
```

```
ub-integer-options INTEGER ::= 256
ub-numeric-user-id-length INTEGER ::= 32
ub-organization-name-length INTEGER ::= 64
ub-organizational-unit-name-length INTEGER ::= 32
ub-organizational-units INTEGER ::= 4
ub-pds-name-length INTEGER ::= 16
ub-pds-parameter-length INTEGER ::= 30
ub-pds-physical-address-lines INTEGER ::= 6
ub-postal-code-length INTEGER ::= 16
ub-surname-length INTEGER ::= 40
ub-terminal-id-length INTEGER ::= 24
ub-unformatted-address-length INTEGER ::= 180
ub-x121-address-length INTEGER ::= 16

-- Note - upper bounds on string types, such as TeletexString, are
-- measured in characters. Excepting PrintableString or IA5String, a
-- significantly greater number of octets will be required to hold
-- such a value. As a minimum, 16 octets or twice the specified
-- upper bound, whichever is the larger, should be allowed for
-- TeletexString. For UTF8String or UniversalString, at least four
-- times the upper bound should be allowed.

END
```

## 9 Module PKIX1Explicit-2009 (From RFC 5912)

```
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)}

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

IMPORTS

Extensions{}, EXTENSION, ATTRIBUTE, SingleAttribute{
    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{}, PUBLIC-KEY, SIGNATURE-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)}

CertExtensions, CrlExtensions, CrlEntryExtensions
    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)}

SignatureAlgs, PublicKeys
    FROM PKIXAlgs-2009
    {iso(1) identified-organization(3) dod(6)
      internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) 56}

SignatureAlgs, PublicKeys
    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)}
```

```
ORAddress
    FROM PKIX-X400Address-2009
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) pkix(7) id-mod(0) id-mod-pkix1-x400address-02(60)};

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

-- PKIX arcs

id-pe OBJECT IDENTIFIER ::= { id-pkix 1 }
-- arc for private certificate extensions

id-qt OBJECT IDENTIFIER ::= { id-pkix 2 }
-- arc for policy qualifier types

id-kp OBJECT IDENTIFIER ::= { id-pkix 3 }
-- arc for extended key purpose OIDs

id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }
-- arc for access descriptors

-- policyQualifierIds for Internet policy qualifiers
id-qt-cps OBJECT IDENTIFIER ::= { id-qt 1 }
-- OID for CPS qualifier

id-qt-unotice OBJECT IDENTIFIER ::= { id-qt 2 }
-- OID for user notice qualifier

-- access descriptor definitions

id-ad-ocsp OBJECT IDENTIFIER ::= { id-ad 1 }
id-ad-caIssuers OBJECT IDENTIFIER ::= { id-ad 2 }
id-ad-timeStamping OBJECT IDENTIFIER ::= { id-ad 3 }
id-ad-caRepository OBJECT IDENTIFIER ::= { id-ad 5 }

-- attribute data types
AttributeType ::= ATTRIBUTE.&id

-- Replaced by SingleAttribute{}
--
-- AttributeTypeAndValue ::= SEQUENCE {
--     type ATTRIBUTE.&id({SupportedAttributes}),
--     value ATTRIBUTE.&Type({SupportedAttributes}{@type}) }
--
-- Suggested naming attributes: Definition of the following
-- information object set may be augmented to meet local
-- requirements. Note that deleting members of the set may
-- prevent interoperability with conforming implementations.
-- All attributes are presented in pairs: the AttributeType
-- followed by the type definition for the corresponding
-- AttributeValue.

-- Arc for standard naming attributes

id-at OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 4 }

-- Naming attributes of type X520name

id-at-name AttributeType ::= { id-at 41 }

at-name ATTRIBUTE ::= { TYPE X520name IDENTIFIED BY id-at-name }

id-at-surname AttributeType ::= { id-at 4 }

at-surname ATTRIBUTE ::= { TYPE X520name IDENTIFIED BY id-at-surname }
```

```
id-at-givenName      AttributeType ::= { id-at 42 }
at-givenName         ATTRIBUTE ::= { TYPE X520name IDENTIFIED BY id-at-givenName }
id-at-initials       AttributeType ::= { id-at 43 }
at-initials          ATTRIBUTE ::= { TYPE X520name IDENTIFIED BY id-at-initials }
id-at-generationQualifier AttributeType ::= { id-at 44 }
at-generationQualifier ATTRIBUTE ::=
    { TYPE X520name IDENTIFIED BY id-at-generationQualifier }
-- Directory string type --
DirectoryString{INTEGER:maxSize} ::= CHOICE {
    teletexString      TeletexString(SIZE (1..maxSize)),
    printableString    PrintableString(SIZE (1..maxSize)),
    bmpString          BMPString(SIZE (1..maxSize)),
    universalString    UniversalString(SIZE (1..maxSize)),
    uTF8String         UTF8String(SIZE (1..maxSize))
}
X520name ::= DirectoryString {ub-name}
-- Naming attributes of type X520CommonName
id-at-commonName     AttributeType ::= { id-at 3 }
at-x520CommonName    ATTRIBUTE ::=
    {TYPE X520CommonName
     IDENTIFIED BY id-at-commonName }
X520CommonName ::= DirectoryString {ub-common-name}
-- Naming attributes of type X520LocalityName
id-at-localityName   AttributeType ::= { id-at 7 }
at-x520LocalityName  ATTRIBUTE ::= {
    TYPE X520LocalityName
    IDENTIFIED BY id-at-localityName }
X520LocalityName ::= DirectoryString {ub-locality-name}
-- Naming attributes of type X520StateOrProvinceName
id-at-stateOrProvinceName AttributeType ::= { id-at 8 }
at-x520StateOrProvinceName ATTRIBUTE ::= {
    TYPE DirectoryString {ub-state-name}
    IDENTIFIED BY id-at-stateOrProvinceName }
X520StateOrProvinceName ::= DirectoryString {ub-state-name}
-- Naming attributes of type X520OrganizationName
id-at-organizationName AttributeType ::= { id-at 10 }
at-x520OrganizationName ATTRIBUTE ::= {
    TYPE DirectoryString {ub-organization-name}
    IDENTIFIED BY id-at-organizationName }
X520OrganizationName ::= DirectoryString {ub-organization-name}
-- Naming attributes of type X520OrganizationalUnitName
id-at-organizationalUnitName AttributeType ::= { id-at 11 }
at-x520OrganizationalUnitName ATTRIBUTE ::= {
    TYPE DirectoryString {ub-organizational-unit-name}
    IDENTIFIED BY id-at-organizationalUnitName }
X520OrganizationalUnitName ::= DirectoryString{ub-organizational-unit-name}
-- Naming attributes of type X520Title
id-at-title          AttributeType ::= { id-at 12 }
```

```
at-x520Title                ATTRIBUTE ::= {
    TYPE DirectoryString { ub-title }
    IDENTIFIED BY id-at-title }

-- Naming attributes of type X520dnQualifier
id-at-dnQualifier           AttributeType ::= { id-at 46 }
at-x520dnQualifier         ATTRIBUTE ::= {
    TYPE PrintableString
    IDENTIFIED BY id-at-dnQualifier }

-- Naming attributes of type X520countryName (digraph from IS 3166)
id-at-countryName          AttributeType ::= { id-at 6 }
at-x520countryName         ATTRIBUTE ::= {
    TYPE PrintableString (SIZE (2))
    IDENTIFIED BY id-at-countryName }

-- Naming attributes of type X520SerialNumber
id-at-serialNumber         AttributeType ::= { id-at 5 }
at-x520SerialNumber        ATTRIBUTE ::= {
    TYPE PrintableString(SIZE (1..ub-serial-number))
    IDENTIFIED BY id-at-serialNumber }

-- Naming attributes of type X520Pseudonym
id-at-pseudonym            AttributeType ::= { id-at 65 }
at-x520Pseudonym           ATTRIBUTE ::= {
    TYPE DirectoryString {ub-pseudonym}
    IDENTIFIED BY id-at-pseudonym }

-- Naming attributes of type DomainComponent (from RFC 2247)
id-domainComponent         AttributeType ::=
    { itu-t(0) data(9) pss(2342) ucl(19200300) pilot(100)
      pilotAttributeType(1) 25 }
at-domainComponent         ATTRIBUTE ::= {
    TYPE IA5String
    IDENTIFIED BY id-domainComponent }

-- Legacy attributes
pkcs-9 OBJECT IDENTIFIER ::=
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 9 }
id-emailAddress            AttributeType ::= { pkcs-9 1 }
at-emailAddress            ATTRIBUTE ::= {
    TYPE IA5String
    (SIZE (1..ub-emailaddress-length))
    IDENTIFIED BY id-emailAddress }

-- naming data types --
Name ::= CHOICE { -- only one possibility for now --
    rdnSequence RDNSequence }

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
DistinguishedName ::= RDNSequence
RelativeDistinguishedName ::= SET SIZE (1 .. MAX) OF
    SingleAttribute { {SupportedAttributes} }

-- These are the known name elements for a DN
SupportedAttributes ATTRIBUTE ::= {
    at-name | at-surname | at-givenName | at-initials |
    at-generationQualifier | at-x520CommonName |
    at-x520LocalityName | at-x520StateOrProvinceName |
```

```
at-x520OrganizationName | at-x520OrganizationalUnitName |  
at-x520Title | at-x520dnQualifier | at-x520countryName |  
at-x520SerialNumber | at-x520Pseudonym | at-domainComponent |  
at-emailAddress,  
...  
}
```

```
--  
-- Certificate- and CRL-specific structures begin here  
--
```

```
Certificate ::= SIGNED{TBSCertificate}
```

```
TBSCertificate ::= SEQUENCE {  
    version                [0] Version DEFAULT v1,  
    serialNumber           CertificateSerialNumber,  
    signature              AlgorithmIdentifier{SIGNATURE-ALGORITHM,  
        {SignatureAlgorithms}},  
    issuer                 Name,  
    validity               Validity,  
    subject                Name,  
    subjectPublicKeyInfo   SubjectPublicKeyInfo,  
    ... ,  
    [[2:                  -- If present, version MUST be v2  
    issuerUniqueID         [1] IMPLICIT UniqueIdentifier OPTIONAL,  
    subjectUniqueID       [2] IMPLICIT UniqueIdentifier OPTIONAL  
    ]],  
    [[3:                  -- If present, version MUST be v3 --  
    extensions             [3] Extensions{{CertExtensions}} OPTIONAL  
    ]],  
    ...  
}
```

```
Version ::= INTEGER { v1(0), v2(1), v3(2) }
```

```
CertificateSerialNumber ::= INTEGER
```

```
Validity ::= SEQUENCE {  
    notBefore      Time,  
    notAfter       Time  
}
```

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

```
UniqueIdentifier ::= BIT STRING
```

```
SubjectPublicKeyInfo ::= SEQUENCE {  
    algorithm      AlgorithmIdentifier{PUBLIC-KEY,  
        {PublicKeyAlgorithms}},  
    subjectPublicKey BIT STRING  
}
```

```
-- CRL structures
```

```
CertificateList ::= SIGNED{TBSCertList}
```

```
TBSCertList ::= SEQUENCE {  
    version                Version OPTIONAL,  
    -- if present, MUST be v2  
    signature              AlgorithmIdentifier{SIGNATURE-ALGORITHM,  
        {SignatureAlgorithms}},  
    issuer                 Name,  
    thisUpdate             Time,  
    nextUpdate             Time OPTIONAL,  
    revokedCertificates    SEQUENCE SIZE (1..MAX) OF SEQUENCE {  
        userCertificate      CertificateSerialNumber,  
        revocationDate      Time,  
        ... ,  
        [[2:                  -- if present, version MUST be v2
```

```
        crlEntryExtensions    Extensions{{CrlEntryExtensions}} OPTIONAL
        ]], ...
    } OPTIONAL,
    ... ,
    [[2:                        -- if present, version MUST be v2
    crlExtensions              [0] Extensions{{CrlExtensions}}
        OPTIONAL
    ]],
    ...
    }

-- Version, Time, CertificateSerialNumber, and Extensions were
-- defined earlier for use in the certificate structure
--
-- The two object sets below should be expanded to include
-- those algorithms which are supported by the system.
--
-- For example:
-- SignatureAlgorithms SIGNATURE-ALGORITHM ::= {
--     PKIXAlgs-2008.SignatureAlgs, ...,
--     - - RFC 3279 provides the base set
--     PKIX1-PSS-OAEP-ALGORITHMS.SignatureAlgs |
--     - - RFC 4055 provides extension algs
--     OtherModule.SignatureAlgs
--     - - RFC XXXX provides additional extension algs
-- }

SignatureAlgorithms SIGNATURE-ALGORITHM ::= {
    PKIXAlgs-2009.SignatureAlgs, ...,
    PKIX1-PSS-OAEP-Algorithms-2009.SignatureAlgs
}

PublicKeyAlgorithms PUBLIC-KEY ::= {
    PKIXAlgs-2009.PublicKeys, ...,
    PKIX1-PSS-OAEP-Algorithms-2009.PublicKeys
}

-- Upper Bounds
ub-state-name INTEGER ::= 128
ub-organization-name INTEGER ::= 64
ub-organizational-unit-name INTEGER ::= 64
ub-title INTEGER ::= 64
ub-serial-number INTEGER ::= 64
ub-pseudonym INTEGER ::= 128
ub-emailaddress-length INTEGER ::= 255
ub-locality-name INTEGER ::= 128
ub-common-name INTEGER ::= 64
ub-name INTEGER ::= 32768

-- Note - upper bounds on string types, such as TeletexString, are
-- measured in characters. Excepting PrintableString or IA5String, a
-- significantly greater number of octets will be required to hold
-- such a value. As a minimum, 16 octets or twice the specified
-- upper bound, whichever is the larger, should be allowed for
-- TeletexString. For UTF8String or UniversalString, at least four
-- times the upper bound should be allowed.
-- Information object classes used in the definition
-- of certificates and CRLs
-- Parameterized Type SIGNED
--
-- Three different versions of doing SIGNED:
-- 1. Simple and close to the previous version
--
```



```
-- SIGNED{ToBeSigned} ::= SEQUENCE {
--   toBeSigned   ToBeSigned,
--   algorithm    AlgorithmIdentifier{SIGNATURE-ALGORITHM,
--                     {SignatureAlgorithms}},
--   signature    BIT STRING
-- }
-- 2. From Authenticated Framework
--
-- SIGNED{ToBeSigned} ::= SEQUENCE {
--   toBeSigned   ToBeSigned,
--   COMPONENTS OF SIGNATURE{ToBeSigned}
-- }
-- SIGNATURE{ToBeSigned} ::= SEQUENCE {
--   algorithmIdentifier AlgorithmIdentifier,
--   encrypted           ENCRYPTED-HASH{ToBeSigned}
-- }
-- ENCRYPTED-HASH{ToBeSigned} ::=
--   BIT STRING
--   (CONSTRAINED BY {
--     shall be the result of applying a hashing procedure to
--     the DER-encoded (see 4.1) octets of a value of
--     ToBeSigned and then applying an encipherment procedure
--     to those octets
--   })
--
-- 3. A more complex version, but one that automatically ties
--     together both the signature algorithm and the
--     signature value for automatic decoding.
--
SIGNED{ToBeSigned} ::= SEQUENCE {
    toBeSigned           ToBeSigned,
    algorithmIdentifier  SEQUENCE {
        algorithm        SIGNATURE-ALGORITHM.&id({SignatureAlgorithms}),
        parameters       SIGNATURE-ALGORITHM.&Params({SignatureAlgorithms}
            {@algorithmIdentifier.algorithm}) OPTIONAL
    },
    signature            BIT STRING
        (CONTAINING SIGNATURE-ALGORITHM.&Value
            ({SignatureAlgorithms}{@algorithmIdentifier.algorithm}))
}

END
```

## 10 Module PKIXImplicit-2009 (From RFC 5912)

```
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)}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

IMPORTS

AttributeSet{}, EXTENSION, 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) }

id-pe, id-kp, id-qt-unotice, id-qt-cps, ORAddress, Name,
RelativeDistinguishedName, CertificateSerialNumber,
DirectoryString{}, SupportedAttributes
    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) };

CertExtensions EXTENSION ::= {
    ext-AuthorityKeyIdentifier | ext-SubjectKeyIdentifier |
    ext-KeyUsage | ext-PrivateKeyUsagePeriod |
    ext-CertificatePolicies | ext-PolicyMappings |
    ext-SubjectAltName | ext-IssuerAltName |
    ext-SubjectDirectoryAttributes |
    ext-BasicConstraints | ext-NameConstraints |
    ext-PolicyConstraints | ext-ExtKeyUsage |
    ext-CRLDistributionPoints | ext-InhibitAnyPolicy |
    ext-FreshestCRL | ext-AuthorityInfoAccess |
    ext-SubjectInfoAccessSyntax,
    ...
}

CrlExtensions EXTENSION ::= {
    ext-AuthorityKeyIdentifier | ext-IssuerAltName |
    ext-CRLNumber | ext-DeltaCRLIndicator |
    ext-IssuingDistributionPoint | ext-FreshestCRL,
    ...
}

CrlEntryExtensions EXTENSION ::= {
    ext-CRLReason | ext-CertificateIssuer |
    ext-HoldInstructionCode | ext-InvalidityDate,
    ...
}

-- Shared arc for standard certificate and CRL extensions
id-ce OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 29 }

-- authority key identifier OID and syntax
ext-AuthorityKeyIdentifier EXTENSION ::= {
    SYNTAX AuthorityKeyIdentifier
    IDENTIFIED BY id-ce-authorityKeyIdentifier }

id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 35 }

AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier [0] KeyIdentifier OPTIONAL,
    authorityCertIssuer [1] GeneralNames OPTIONAL,
    authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL }
    (WITH COMPONENTS {
        .../
        authorityCertIssuer PRESENT,
        authorityCertSerialNumber PRESENT
    } |
    WITH COMPONENTS {
        .../
        authorityCertIssuer ABSENT,
        authorityCertSerialNumber ABSENT
    })

KeyIdentifier ::= OCTET STRING

-- subject key identifier OID and syntax
ext-SubjectKeyIdentifier EXTENSION ::= {
    SYNTAX KeyIdentifier
    IDENTIFIED BY id-ce-subjectKeyIdentifier }

id-ce-subjectKeyIdentifier OBJECT IDENTIFIER ::= { id-ce 14 }

-- key usage extension OID and syntax
ext-KeyUsage EXTENSION ::= {
    SYNTAX KeyUsage
    IDENTIFIED BY id-ce-keyUsage }

id-ce-keyUsage OBJECT IDENTIFIER ::= { id-ce 15 }
```

```
KeyUsage ::= BIT STRING {
    digitalSignature (0),
    nonRepudiation (1), -- recent editions of X.509 have
    -- renamed this bit to
    -- contentCommitment
    keyEncipherment (2),
    dataEncipherment (3),
    keyAgreement (4),
    keyCertSign (5),
    cRLSign (6),
    encipherOnly (7),
    decipherOnly (8)
}

-- private key usage period extension OID and syntax
ext-PrivateKeyUsagePeriod EXTENSION ::= {
    SYNTAX PrivateKeyUsagePeriod
    IDENTIFIED BY id-ce-privateKeyUsagePeriod }

id-ce-privateKeyUsagePeriod OBJECT IDENTIFIER ::= { id-ce 16 }

PrivateKeyUsagePeriod ::= SEQUENCE {
    notBefore [0] GeneralizedTime OPTIONAL,
    notAfter [1] GeneralizedTime OPTIONAL }
(WITH COMPONENTS {..., notBefore PRESENT } |
WITH COMPONENTS {..., notAfter PRESENT })

-- certificate policies extension OID and syntax
ext-CertificatePolicies EXTENSION ::= {
    SYNTAX CertificatePolicies
    IDENTIFIED BY id-ce-certificatePolicies}

id-ce-certificatePolicies OBJECT IDENTIFIER ::= { id-ce 32 }

CertificatePolicies ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation

PolicyInformation ::= SEQUENCE {
    policyIdentifier CertPolicyId,
    policyQualifiers SEQUENCE SIZE (1..MAX) OF PolicyQualifierInfo
    OPTIONAL }

CertPolicyId ::= OBJECT IDENTIFIER

CERT-POLICY-QUALIFIER ::= TYPE-IDENTIFIER

PolicyQualifierInfo ::= SEQUENCE {
    policyQualifierId CERT-POLICY-QUALIFIER.&id({PolicyQualifierId}),
    qualifier CERT-POLICY-QUALIFIER.&Type
    ({PolicyQualifierId}@policyQualifierId)}

-- Implementations that recognize additional policy qualifiers MUST
-- augment the following definition for PolicyQualifierId

PolicyQualifierId CERT-POLICY-QUALIFIER ::=
    { pqid-cps | pqid-unotice, ... }

pqid-cps CERT-POLICY-QUALIFIER ::= { CPSuri IDENTIFIED BY id-qt-cps }

pqid-unotice CERT-POLICY-QUALIFIER ::= {
    UserNotice
    IDENTIFIED BY id-qt-unotice }

-- CPS pointer qualifier
CPSuri ::= IA5String

-- user notice qualifier
UserNotice ::= SEQUENCE {
    noticeRef NoticeReference OPTIONAL,
    explicitText DisplayText OPTIONAL
}
```

```
--
-- This is not made explicit in the text
--
-- {WITH COMPONENTS {..., noticeRef PRESENT} |
-- WITH COMPONENTS {..., DisplayText PRESENT }}

NoticeReference ::= SEQUENCE {
    organization      DisplayText,
    noticeNumbers     SEQUENCE OF INTEGER
}

DisplayText ::= CHOICE {
    ia5String          IA5String      (SIZE (1..200)),
    visibleString      VisibleString (SIZE (1..200)),
    bmpString          BMPString      (SIZE (1..200)),
    utf8String         UTF8String     (SIZE (1..200))
}

-- policy mapping extension OID and syntax
ext-PolicyMappings EXTENSION ::= {
    SYNTAX PolicyMappings IDENTIFIED BY id-ce-policyMappings }

id-ce-policyMappings OBJECT IDENTIFIER ::= { id-ce 33 }

PolicyMappings ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
    issuerDomainPolicy      CertPolicyId,
    subjectDomainPolicy     CertPolicyId
}

-- subject alternative name extension OID and syntax
ext-SubjectAltName EXTENSION ::= {
    SYNTAX GeneralNames
    IDENTIFIED BY id-ce-subjectAltName }

id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
    otherName           [0] INSTANCE OF OTHER-NAME,
    rfc822Name          [1] IA5String,
    dNSName             [2] IA5String,
    x400Address         [3] ORAddress,
    directoryName       [4] Name,
    ediPartyName        [5] EDIPartyName,
    uniformResourceIdentifier [6] IA5String,
    iPAddress           [7] OCTET STRING,
    registeredID        [8] OBJECT IDENTIFIER
}

-- AnotherName replaces OTHER-NAME ::= TYPE-IDENTIFIER, as
-- TYPE-IDENTIFIER is not supported in the '88 ASN.1 syntax
OTHER-NAME ::= TYPE-IDENTIFIER

EDIPartyName ::= SEQUENCE {
    nameAssigner        [0] DirectoryString {ubMax} OPTIONAL,
    partyName           [1] DirectoryString {ubMax}
}

-- issuer alternative name extension OID and syntax
ext-IssuerAltName EXTENSION ::= {
    SYNTAX GeneralNames
    IDENTIFIED BY id-ce-issuerAltName }

id-ce-issuerAltName OBJECT IDENTIFIER ::= { id-ce 18 }

ext-SubjectDirectoryAttributes EXTENSION ::= {
    SYNTAX SubjectDirectoryAttributes
    IDENTIFIED BY id-ce-subjectDirectoryAttributes }
```

```
id-ce-subjectDirectoryAttributes OBJECT IDENTIFIER ::= { id-ce 9 }
SubjectDirectoryAttributes ::= SEQUENCE SIZE (1..MAX) OF
    AttributeSet{{SupportedAttributes}}

-- basic constraints extension OID and syntax
ext-BasicConstraints EXTENSION ::= {
    SYNTAX BasicConstraints
    IDENTIFIED BY id-ce-basicConstraints }

id-ce-basicConstraints OBJECT IDENTIFIER ::= { id-ce 19 }
BasicConstraints ::= SEQUENCE {
    cA                BOOLEAN DEFAULT FALSE,
    pathLenConstraint INTEGER (0..MAX) OPTIONAL
}

-- name constraints extension OID and syntax
ext-NameConstraints EXTENSION ::= {
    SYNTAX NameConstraints
    IDENTIFIED BY id-ce-nameConstraints }

id-ce-nameConstraints OBJECT IDENTIFIER ::= { id-ce 30 }
NameConstraints ::= SEQUENCE {
    permittedSubtrees [0] GeneralSubtrees OPTIONAL,
    excludedSubtrees  [1] GeneralSubtrees OPTIONAL
}

--
-- This is a constraint in the issued certificates by CAs, but is
-- not a requirement on EEs.
--
-- (WITH COMPONENTS { ..., permittedSubtrees PRESENT} |
-- WITH COMPONENTS { ..., excludedSubtrees PRESENT })
GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree
GeneralSubtree ::= SEQUENCE {
    base            GeneralName,
    minimum         [0] BaseDistance DEFAULT 0,
    maximum         [1] BaseDistance OPTIONAL
}

BaseDistance ::= INTEGER (0..MAX)

-- policy constraints extension OID and syntax
ext-PolicyConstraints EXTENSION ::= {
    SYNTAX PolicyConstraints
    IDENTIFIED BY id-ce-policyConstraints }

id-ce-policyConstraints OBJECT IDENTIFIER ::= { id-ce 36 }
PolicyConstraints ::= SEQUENCE {
    requireExplicitPolicy [0] SkipCerts OPTIONAL,
    inhibitPolicyMapping  [1] SkipCerts OPTIONAL
}

--
-- This is a constraint in the issued certificates by CAs,
-- but is not a requirement for EEs
--
-- (WITH COMPONENTS { ..., requireExplicitPolicy PRESENT} |
-- WITH COMPONENTS { ..., inhibitPolicyMapping PRESENT})
SkipCerts ::= INTEGER (0..MAX)

-- CRL distribution points extension OID and syntax
ext-CRLDistributionPoints EXTENSION ::= {
    SYNTAX CRLDistributionPoints
    IDENTIFIED BY id-ce-cRLDistributionPoints}
```

```
id-ce-cRLDistributionPoints OBJECT IDENTIFIER ::= {id-ce 31}
CRLDistributionPoints ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint
DistributionPoint ::= SEQUENCE {
    distributionPoint [0] DistributionPointName OPTIONAL,
    reasons           [1] ReasonFlags OPTIONAL,
    cRLIssuer         [2] GeneralNames OPTIONAL
}

--
-- This is not a requirement in the text, but it seems as if it
-- should be
--
--(WITH COMPONENTS {..., distributionPoint PRESENT} |
-- WITH COMPONENTS {..., cRLIssuer PRESENT})

DistributionPointName ::= CHOICE {
    fullName           [0] GeneralNames,
    nameRelativeToCRLIssuer [1] RelativeDistinguishedName
}

ReasonFlags ::= BIT STRING {
    unused           (0),
    keyCompromise   (1),
    cACompromise    (2),
    affiliationChanged (3),
    superseded      (4),
    cessationOfOperation (5),
    certificateHold  (6),
    privilegeWithdrawn (7),
    aACompromise    (8)
}

-- extended key usage extension OID and syntax
ext-ExtKeyUsage EXTENSION ::= {
    SYNTAX ExtKeyUsageSyntax
    IDENTIFIED BY id-ce-extKeyUsage }

id-ce-extKeyUsage OBJECT IDENTIFIER ::= {id-ce 37}
ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId
KeyPurposeId ::= OBJECT IDENTIFIER

-- permit unspecified key uses
anyExtendedKeyUsage OBJECT IDENTIFIER ::= { id-ce-extKeyUsage 0 }

-- extended key purpose OIDs
id-kp-serverAuth OBJECT IDENTIFIER ::= { id-kp 1 }
id-kp-clientAuth OBJECT IDENTIFIER ::= { id-kp 2 }
id-kp-codeSigning OBJECT IDENTIFIER ::= { id-kp 3 }
id-kp-emailProtection OBJECT IDENTIFIER ::= { id-kp 4 }
id-kp-timeStamping OBJECT IDENTIFIER ::= { id-kp 8 }
id-kp-OCSPSigning OBJECT IDENTIFIER ::= { id-kp 9 }

-- inhibit any policy OID and syntax
ext-InhibitAnyPolicy EXTENSION ::= {
    SYNTAX SkipCerts
    IDENTIFIED BY id-ce-inhibitAnyPolicy }

id-ce-inhibitAnyPolicy OBJECT IDENTIFIER ::= { id-ce 54 }

-- freshest (delta)CRL extension OID and syntax
ext-FreshestCRL EXTENSION ::= {
    SYNTAX CRLDistributionPoints
```

```
IDENTIFIED BY id-ce-freshestCRL }
id-ce-freshestCRL OBJECT IDENTIFIER ::= { id-ce 46 }
-- authority info access
ext-AuthorityInfoAccess EXTENSION ::= {
    SYNTAX AuthorityInfoAccessSyntax
    IDENTIFIED BY id-pe-authorityInfoAccess }
id-pe-authorityInfoAccess OBJECT IDENTIFIER ::= { id-pe 1 }
AuthorityInfoAccessSyntax ::= SEQUENCE SIZE (1..MAX) OF AccessDescription
AccessDescription ::= SEQUENCE {
    accessMethod      OBJECT IDENTIFIER,
    accessLocation    GeneralName
}
-- subject info access
ext-SubjectInfoAccessSyntax EXTENSION ::= {
    SYNTAX SubjectInfoAccessSyntax
    IDENTIFIED BY id-pe-subjectInfoAccess }
id-pe-subjectInfoAccess OBJECT IDENTIFIER ::= { id-pe 11 }
SubjectInfoAccessSyntax ::= SEQUENCE SIZE (1..MAX) OF AccessDescription
-- CRL number extension OID and syntax
ext-CRLNumber EXTENSION ::= {
    SYNTAX INTEGER (0..MAX)
    IDENTIFIED BY id-ce-cRLNumber }
id-ce-cRLNumber OBJECT IDENTIFIER ::= { id-ce 20 }
CRLNumber ::= INTEGER (0..MAX)
-- issuing distribution point extension OID and syntax
ext-IssuingDistributionPoint EXTENSION ::= {
    SYNTAX IssuingDistributionPoint
    IDENTIFIED BY id-ce-issuingDistributionPoint }
id-ce-issuingDistributionPoint OBJECT IDENTIFIER ::= { id-ce 28 }
IssuingDistributionPoint ::= SEQUENCE {
    distributionPoint                [0] DistributionPointName OPTIONAL,
    onlyContainsUserCerts             [1] BOOLEAN DEFAULT FALSE,
    onlyContainsCACerts               [2] BOOLEAN DEFAULT FALSE,
    onlySomeReasons                   [3] ReasonFlags OPTIONAL,
    indirectCRL                       [4] BOOLEAN DEFAULT FALSE,
    onlyContainsAttributeCerts        [5] BOOLEAN DEFAULT FALSE
}
-- at most one of onlyContainsUserCerts, onlyContainsCACerts,
-- or onlyContainsAttributeCerts may be set to TRUE.
ext-DeltaCRLIndicator EXTENSION ::= {
    SYNTAX CRLNumber
    IDENTIFIED BY id-ce-deltaCRLIndicator }
id-ce-deltaCRLIndicator OBJECT IDENTIFIER ::= { id-ce 27 }
-- CRL reasons extension OID and syntax
ext-CRLReason EXTENSION ::= {
    SYNTAX CRLReason
    IDENTIFIED BY id-ce-cRLReasons }
id-ce-cRLReasons OBJECT IDENTIFIER ::= { id-ce 21 }
CRLReason ::= ENUMERATED {
    unspecified                (0),
    keyCompromise               (1),
```

```

    cACompromise          (2),
    affiliationChanged     (3),
    superseded            (4),
    cessationOfOperation  (5),
    certificateHold       (6),
    removeFromCRL         (8),
    privilegeWithdrawn    (9),
    aACompromise          (10)
}

-- certificate issuer CRL entry extension OID and syntax
ext-CertificateIssuer EXTENSION ::= {
    SYNTAX GeneralNames
    IDENTIFIED BY id-ce-certificateIssuer }

id-ce-certificateIssuer OBJECT IDENTIFIER ::= { id-ce 29 }

-- hold instruction extension OID and syntax
ext-HoldInstructionCode EXTENSION ::= {
    SYNTAX OBJECT IDENTIFIER
    IDENTIFIED BY id-ce-holdInstructionCode }

id-ce-holdInstructionCode OBJECT IDENTIFIER ::= { id-ce 23 }

-- ANSI x9 holdinstructions
holdInstruction OBJECT IDENTIFIER ::=
    {joint-iso-itu-t(2) member-body(2) us(840) x9cm(10040) 2}

id-holdinstruction-none OBJECT IDENTIFIER ::= {holdInstruction 1} -- deprecated
id-holdinstruction-callissuer OBJECT IDENTIFIER ::= {holdInstruction 2}
id-holdinstruction-reject OBJECT IDENTIFIER ::= {holdInstruction 3}

-- invalidity date CRL entry extension OID and syntax
ext-InvalidityDate EXTENSION ::= {
    SYNTAX GeneralizedTime
    IDENTIFIED BY id-ce-invalidityDate }

id-ce-invalidityDate OBJECT IDENTIFIER ::= { id-ce 24 }

-- Upper bounds
ubMax INTEGER ::= 32768

END
```

## 11 Module PKIX1-PSS-OAEP-Algorithms-2009 (From RFC 5912)

```

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)}

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

IMPORTS

AlgorithmIdentifier {}, ALGORITHM, DIGEST-ALGORITHM, KEY-TRANSPORT,
SIGNATURE-ALGORITHM, PUBLIC-KEY, 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)}

id-sha1, mda-sha1, pk-rsa, RSAPublicKey
    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)};
```



```
-- =====
--   Object Set exports
-- =====
--
-- Define top-level symbols with all of the objects defined for
-- export to other modules.  These objects would be included as part
-- of an Object Set to restrict the set of legal values.
--
PublicKeys PUBLIC-KEY ::= { pk-rsaSSA-PSS | pk-rsaES-OAEP, ... }
SignatureAlgs SIGNATURE-ALGORITHM ::= { sa-rsaSSA-PSS, ...}
KeyTransportAlgs KEY-TRANSPORT ::= { kta-rsaES-OAEP, ... }
HashAlgs DIGEST-ALGORITHM ::= { mda-sha224 | mda-sha256 | mda-sha384 | mda-sha512,
    ...
}
SMimeCaps SMIME-CAPS ::= {
    sa-rsaSSA-PSS.&smimeCaps |
    kta-rsaES-OAEP.&smimeCaps,
    ...
}
-- =====
--   Algorithm Objects
-- =====
--
-- Public key object for PSS signatures
pk-rsaSSA-PSS PUBLIC-KEY ::= {
    IDENTIFIER id-RSASSA-PSS
    KEY RSAPublicKey
    PARAMS TYPE RSASSA-PSS-params ARE optional
    -- Private key format not in this module --
    CERT-KEY-USAGE { nonRepudiation, digitalSignature,
    keyCertSign, cRLSign }
}
--
-- Signature algorithm definition for PSS signatures
--
sa-rsaSSA-PSS SIGNATURE-ALGORITHM ::= {
    IDENTIFIER id-RSASSA-PSS
    PARAMS TYPE RSASSA-PSS-params ARE required
    HASHES { mda-sha1 | mda-sha224 | mda-sha256 | mda-sha384
    | mda-sha512 }
    PUBLIC-KEYS { pk-rsa | pk-rsaSSA-PSS }
    SMIME-CAPS { IDENTIFIED BY id-RSASSA-PSS }
}
--
-- Signature algorithm definitions for PKCS v1.5 signatures
--
sa-sha224WithRSAEncryption SIGNATURE-ALGORITHM ::= {
    IDENTIFIER sha224WithRSAEncryption
    PARAMS TYPE NULL ARE required
    HASHES { mda-sha224 }
    PUBLIC-KEYS { pk-rsa }
    SMIME-CAPS { IDENTIFIED BY sha224WithRSAEncryption }
}
sha224WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 14 }
sa-sha256WithRSAEncryption SIGNATURE-ALGORITHM ::= {
    IDENTIFIER sha256WithRSAEncryption
    PARAMS TYPE NULL ARE required
    HASHES { mda-sha256 }
    PUBLIC-KEYS { pk-rsa }
}
```

```
SMIME-CAPS { IDENTIFIED BY sha256WithRSAEncryption }
}

sha256WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 11 }

sa-sha384WithRSAEncryption SIGNATURE-ALGORITHM ::= {
    IDENTIFIER sha384WithRSAEncryption
    PARAMS TYPE NULL ARE required
    HASHES { mda-sha384 }
    PUBLIC-KEYS { pk-rsa }
    SMIME-CAPS { IDENTIFIED BY sha384WithRSAEncryption }
}

sha384WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 12 }

sa-sha512WithRSAEncryption SIGNATURE-ALGORITHM ::= {
    IDENTIFIER sha512WithRSAEncryption
    PARAMS TYPE NULL ARE required
    HASHES { mda-sha512 }
    PUBLIC-KEYS { pk-rsa }
    SMIME-CAPS { IDENTIFIED BY sha512WithRSAEncryption }
}

sha512WithRSAEncryption OBJECT IDENTIFIER ::= { pkcs-1 13 }

--
-- Public key definition for OAEP encryption
--

pk-rsaES-OAEP PUBLIC-KEY ::= {
    IDENTIFIER id-RSAES-OAEP
    KEY RSAPublicKey
    PARAMS TYPE RSAES-OAEP-params ARE optional
    -- Private key format not in this module --
    CERT-KEY-USAGE {keyEncipherment, dataEncipherment}
}

--
-- Key transport key lock definition for OAEP encryption
--

kta-rsaES-OAEP KEY-TRANSPORT ::= {
    IDENTIFIER id-RSAES-OAEP
    PARAMS TYPE RSAES-OAEP-params ARE required
    PUBLIC-KEYS { pk-rsa | pk-rsaES-OAEP }
    SMIME-CAPS { TYPE RSAES-OAEP-params IDENTIFIED BY id-RSAES-OAEP}
}

-- =====
-- Basic object identifiers
-- =====

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

-- When rsaEncryption is used in an AlgorithmIdentifier, the
-- parameters MUST be present and MUST be NULL.
-- rsaEncryption OBJECT IDENTIFIER ::= { pkcs-1 1 }
-- When id-RSAES-OAEP is used in an AlgorithmIdentifier,
-- and the parameters field is present, it MUST be
-- RSAES-OAEP-params.

id-RSAES-OAEP OBJECT IDENTIFIER ::= { pkcs-1 7 }

-- When id-mgf1 is used in an AlgorithmIdentifier, the parameters
-- MUST be present and MUST be a HashAlgorithm.

id-mgf1 OBJECT IDENTIFIER ::= { pkcs-1 8 }

-- When id-pSpecified is used in an AlgorithmIdentifier, the
-- parameters MUST be an OCTET STRING.

id-pSpecified OBJECT IDENTIFIER ::= { pkcs-1 9 }
```

```
-- When id-RSASSA-PSS is used in an AlgorithmIdentifier, and the
-- parameters field is present, it MUST be RSASSA-PSS-params.
id-RSASSA-PSS OBJECT IDENTIFIER ::= { pkcs-1 10 }

-- When the following OIDs are used in an AlgorithmIdentifier, the
-- parameters SHOULD be absent, but if the parameters are present,
-- they MUST be NULL.
--
-- id-sha1 is imported from RFC 3279.  Additionally, the v1.5
-- signature algorithms (i.e., rsaWithSHA256) are now solely placed
-- in that module.
--
id-sha224 OBJECT IDENTIFIER ::=
    { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
      csor(3) nistAlgorithms(4) hashalgs(2) 4 }

mda-sha224 DIGEST-ALGORITHM ::= {
    IDENTIFIER id-sha224
    PARAMS TYPE NULL ARE preferredAbsent
}

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

mda-sha256 DIGEST-ALGORITHM ::= {
    IDENTIFIER id-sha256
    PARAMS TYPE NULL ARE preferredAbsent
}

id-sha384 OBJECT IDENTIFIER ::=
    { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
      csor(3) nistAlgorithms(4) hashalgs(2) 2 }

mda-sha384 DIGEST-ALGORITHM ::= {
    IDENTIFIER id-sha384
    PARAMS TYPE NULL ARE preferredAbsent
}

id-sha512 OBJECT IDENTIFIER ::=
    { joint-iso-itu-t(2) country(16) us(840) organization(1) gov(101)
      csor(3) nistAlgorithms(4) hashalgs(2) 3 }

mda-sha512 DIGEST-ALGORITHM ::= {
    IDENTIFIER id-sha512
    PARAMS TYPE NULL ARE preferredAbsent
}

-- =====
-- Constants
-- =====

EncodingParameters ::= OCTET STRING(SIZE(0..MAX))

nullOctetString EncodingParameters ::= 'H

nullParameters NULL ::= NULL

-- =====
-- Algorithm Identifiers
-- =====

HashAlgorithm ::= AlgorithmIdentifier{DIGEST-ALGORITHM, {HashAlgorithms}}

HashAlgorithms DIGEST-ALGORITHM ::= {
    { IDENTIFIER id-sha1 PARAMS TYPE NULL ARE preferredPresent } |
    { IDENTIFIER id-sha224 PARAMS TYPE NULL ARE preferredPresent } |
    { IDENTIFIER id-sha256 PARAMS TYPE NULL ARE preferredPresent } |
    { IDENTIFIER id-sha384 PARAMS TYPE NULL ARE preferredPresent } |
    { IDENTIFIER id-sha512 PARAMS TYPE NULL ARE preferredPresent }
}

sha1Identifier HashAlgorithm ::= {
```

```
        algorithm id-sha1,
        parameters NULL : NULL
    }

--
-- We have a default algorithm - create the value here
--

MaskGenAlgorithm ::= AlgorithmIdentifier{ALGORITHM,{PKCS1MGFAlgorithms}}

mgf1SHA1 MaskGenAlgorithm ::= {
    algorithm id-mgf1,
    parameters HashAlgorithm : sha1Identifier
}

--
-- Define the set of mask generation functions
--
-- If the identifier is id-mgf1, any of the listed hash
-- algorithms may be used.
--

PKCS1MGFAlgorithms ALGORITHM ::= {
    { IDENTIFIER id-mgf1 PARAMS TYPE HashAlgorithm ARE required },
    ...
}

--
-- Define the set of known source algorithms for PSS
--

PSourceAlgorithm ::= AlgorithmIdentifier{ALGORITHM,{PSS-SourceAlgorithms}}

PSS-SourceAlgorithms ALGORITHM ::= {
    { IDENTIFIER id-pSpecified PARAMS TYPE EncodingParameters
    ARE required },
    ...
}

pSpecifiedEmpty PSourceAlgorithm ::= {
    algorithm id-pSpecified,
    parameters EncodingParameters : nullOctetString
}

-- =====
-- Main structures
-- =====
-- AlgorithmIdentifier parameters for id-RSASSA-PSS.
-- Note that the tags in this Sequence are explicit.
-- Note: The hash algorithm in hashAlgorithm and in
-- maskGenAlgorithm should be the same.

RSASSA-PSS-params ::= SEQUENCE {
    hashAlgorithm      [0] HashAlgorithm DEFAULT sha1Identifier,
    maskGenAlgorithm   [1] MaskGenAlgorithm DEFAULT mgf1SHA1,
    saltLength         [2] INTEGER DEFAULT 20,
    trailerField       [3] INTEGER DEFAULT 1
}

-- AlgorithmIdentifier parameters for id-RSAES-OAEP.
-- Note that the tags in this Sequence are explicit.
-- Note: The hash algorithm in hashFunc and in
-- maskGenFunc should be the same.

RSAES-OAEP-params ::= SEQUENCE {
    hashFunc           [0] HashAlgorithm DEFAULT sha1Identifier,
    maskGenFunc        [1] MaskGenAlgorithm DEFAULT mgf1SHA1,
    pSourceFunc        [2] PSourceAlgorithm DEFAULT
    pSpecifiedEmpty
}
```

END

## 12 Module SecureMimeMessageV3dot1-2009 (From RFC 5911)

```
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-2010
    {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) id-mod-cms-2009(58) }

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-pkalg-
      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,
    receiptKeyId             [1] RecipientKeyIdentifier,
    subjectAltKeyIdentifier  [2] SubjectKeyIdentifier
}

-- receiptKeyId 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
```

### 13 Module CMSSigncryption

```
CMSSigncryption
{itu-t recommendation(0) x(24) cms-profile(894) module(0) signcryption(0) version1(1)}
"/ITU-T/Recommendation/X/CMS-Profile/Module/Signcryption/Version1"
DEFINITIONS AUTOMATIC TAGS ::=
BEGIN
IMPORTS
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)}
```

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)}

Attribute {}, Attributes, CONTENT-TYPE, EncryptedContentInfo, SignatureAlgorithmIdentifier,  
SignatureValue, SignedAttributes, SignedAttributesSet, SignerIdentifier,  
UnprotectedEncAttributes  
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) }

SigncryptionMechanism  
FROM Signcryption  
{iso(1) standard(0) signcryption(29150) asn1-module(0)  
signcryption-mechanisms(0) version(1)}

signcrypted-attributes, signcrypted-components, signcrypted-content,  
signcrypted-envelope, xPath  
FROM CMSObjectIdentifiers  
{iso(1) identified-organization(3) tc68(133) country(16)  
x9(840) x9Standards(9) x9-73(73) module(0) oids(1) v2009(1)};

id-signcryptedData OBJECT IDENTIFIER ::=   
{itu-t recommendation(0) x(24) cms-profile(894) signcryption(0) data(0)}

ct-SigncryptedData CONTENT-TYPE ::= {  
TYPE SigncryptedData IDENTIFIED BY id-signcryptedData}

SigncryptedData ::= SEQUENCE {  
version CMSVersion,  
contentInformation ContentInformation,  
certificates Certificates OPTIONAL,  
crls CRLs OPTIONAL,  
signcrypters Signcrypters  
}

CMSVersion ::= INTEGER {v0(0)} (0..MAX)

Certificates ::= [XER:BASE64] OCTET STRING

CRLs ::= [XER:BASE64] OCTET STRING

ContentInformation ::= SEQUENCE {  
mode Mode,  
content Content OPTIONAL  
}

Mode ::= MODE.&id({ProcessingModes})

ProcessingModes MODE ::= {  
signcryptedAttributes |  
signcryptedComponents |  
signcryptedContent |  
signcryptedEnveloped,  
... -- Expect additional processing modes --  
}

Content ::= OCTET STRING (SIZE(1..MAX))

NamedKeyEncryptedData ::= SEQUENCE {  
version CMSVersion,  
keyName [0] OCTET STRING OPTIONAL,  
encryptedContentInfo EncryptedContentInfo,  
unprotectedAttrs [1] IMPLICIT Attributes  
{UnprotectedEncAttributes} OPTIONAL  
}

Signcrypters ::= SEQUENCE (SIZE(1..MAX)) OF Signcrypter

Signcrypter ::= SEQUENCE {

```
version          CMSVersion,
side             SigncrypterIDs,
signcryptDataAlgorithm SigncryptDataAlgorithmIdentifier,
signatureValue  SigncryptValue,
signatureInformation SignatureInformation OPTIONAL,
unsigncryptAttributes UnSigncryptAttributes OPTIONAL
}

SigncrypterIDs ::= SEQUENCE {
    sender      KeyPairIdentifier,
    recipient   KeyPairIdentifier
}

KeyPairIdentifier ::= SignerIdentifier

ToBeSigncrypt ::= SEQUENCE {
    content      Content,
    attributes   SigncryptAttributes
}

SigncryptAttributes ::=
    SEQUENCE (SIZE(1..MAX)) OF Attribute{{SigncryptAttributes}}

SigncryptAttributes ATTRIBUTE ::= {
    SignedAttributesSet      | -- CMS Signed Attributes --
    signcryptEnvelope,
    ... -- Expect user defined attributes --
}

signcryptEnvelope ATTRIBUTE ::= {
    TYPE SigncryptKey IDENTIFIED BY signcrypt-envelope
}

SigncryptKey ::= OCTET STRING

SigncryptDataAlgorithmIdentifier ::=
    AlgorithmIdentifier{ALGORITHM, {SigncryptAlgorithms}}

SigncryptAlgorithms ALGORITHM ::= {
    SigncryptMechanism, -- ISO/IEC 29150 Signcrypt --
    ... -- Expect additional algorithm objects --
}

SigncryptValue ::= OCTET STRING(SIZE(1..MAX))

SignatureInformation ::= SEQUENCE {
    signerIdentifier      SignerIdentifier OPTIONAL,
    signatureAlgorithm    SignatureAlgorithmIdentifier OPTIONAL,
    toBeSigned            ToBeSigned,
    signatureValue        SignatureValue
}

ToBeSigned ::= SEQUENCE {
    signcryptPartsManifest SigncryptPartsManifest,
    signedAttributes       SignedAttributes
}

SigncryptPartsManifest ::= Signcrypt{{Manifest}}

Manifest SIGNCRYPTED ::= {
    XPathManifest,
    ... -- Expect additional manifest types --
}

XPathManifest SIGNCRYPTED ::= {
    OID XPath PARMS XPathSet
}

XPathSet ::= SEQUENCE (SIZE(1..MAX)) OF XPath

XPath ::= UTF8String(CONSTRAINED BY { -- XML Path Language 2.0 --})

UnSigncryptAttributes ::=
    SEQUENCE (SIZE(1..MAX)) OF Attribute{{UnSigncryptAttributes}}
```



```
UnSigncryptionAttributes ATTRIBUTE ::= {
... -- Expect additional attributes --
}

-- SigncryptedData processing modes --
signcryptedAttributes MODE ::= {ID signcrypted-attributes}
signcryptedComponents MODE ::= {ID signcrypted-components}
signcryptedContent MODE ::= {ID signcrypted-content}
signcryptedEnveloped      MODE ::= {ID signcrypted-envelope}

-- Information object class and parameterized type definitions --
SIGNCRYPTED ::= CLASS {
    &id OBJECT IDENTIFIER UNIQUE,
    &Type OPTIONAL
}

WITH SYNTAX {OID &id [PARMS &Type]}
Signcrypted{SIGNCRYPTED:IOSet} ::= SEQUENCE {
    name SIGNCRYPTED.&id({IOSet}),
    parts SIGNCRYPTED.&Type({IOSet}{@name}) OPTIONAL
}

MODE ::= CLASS {
    &Type OPTIONAL,
    &id OBJECT IDENTIFIER UNIQUE
}

WITH SYNTAX { [WITH SYNTAX &Type] ID &id}

END
```

## 14 Module CMSCKMKeyManagement

### CMSCKMKeyManagement

```
{itu-t recommendation(0) x(24) cms-profile(894) module(0)
    cKMKeyManagement(1) version1(1)}
```

```
"/ITU-T/Recommendation/X/CMS-Profile/Module/CKMKeyManagement/Version1"
```

```
DEFINITIONS ::=
```

```
BEGIN
```

```
IMPORTS
```

```
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)}
```

```
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)}
```

```
ContentEncryptionAlgorithmIdentifier, DigestAlgorithmIdentifier,
    EncryptedKey, UserKeyingMaterial
    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)}
```

```
id-ckm-key-agree-hash, id-ckm-key-agree-multiple-encrypt,
    id-ckm-symmetric, id-ckm-key-transport, id-ckm-recipient-info
    FROM CMSObjectIdentifiers
    {iso(1) identified-organization(3) tc68(133) country(16) x9(840)
    x9Standards(9) x9-73(73) module(0) oids(1) v2009(1)}
```

```
DomainParameters
```

```
FROM ANSI-X9-42
{iso(1) member-body(2) us(840) ansi-x942(10046) module(5) 1}

ECPParameters
FROM ANSI-X9-62
{iso(1) member-body(2) us(840) 10054 modules(4) 2};

EXTENDED-KEY-MGMT-INFO ::= TYPE-IDENTIFIER

ckmRecipientInfo EXTENDED-KEY-MGMT-INFO ::= {
    KeyConstructRecipientInfo IDENTIFIED BY id-ckm-recipient-info
}

KeyConstructRecipientInfo ::= SEQUENCE {
    version                Version,
    did                    KeyConstructionDomain OPTIONAL,
    ckmid                  [0] KeyConstructionRecipient,
    ukm                    UserKeyingMaterial OPTIONAL,
    keyConstructionAlgorithm KeyConstructionAlgorithmIdentifier,
    encryptedRandom        EncryptedKey
}

Version ::= INTEGER(1..MAX)

KeyConstructionDomain ::= SEQUENCE {
    domainName                DomainName,
    domainMaintenanceLevel    DomainMaintenanceLevel,
    domainParams              DomainParams OPTIONAL
    -- From ANS X9.42 and ANS X.9.62 --
}

DomainParams ::= CHOICE {
    dhParams    [0] DomainParameters,
    ecParams    [1] ECPParameters
}

DomainName ::= PrintableString (SIZE(1..MAX))

DomainMaintenanceLevel ::= INTEGER (1..MAX)

KeyConstructionLabels ::= SEQUENCE SIZE(1..MAX) OF KeyConstructionLabel

KeyConstructionLabel ::= INTEGER

KeyConstructionRecipient ::= CHOICE {
    unencrypted [0] KeyConstructionLabels,
    encrypted   [1] EncryptedRecipientID
}

EncryptedRecipientID ::= SEQUENCE {
    algorithm    ContentEncryptionAlgorithmIdentifier OPTIONAL,
    keyID        [0] OCTET STRING OPTIONAL,
    ciphertext   OCTET STRING
}

KeyConstructionAlgorithmIdentifier ::=
    AlgorithmIdentifier {ALGORITHM, {KeyConstructionAlgorithms}}

KeyConstructionAlgorithms ALGORITHM ::= {
    symmetricConstruction          |
    keyTransportConstruction       |
    keyAgreeMultipleEncryptionConstruction |
    keyAgreeHashConstruction,
    ... -- Expect additional objects --
}

symmetricConstruction    ALGORITHM ::= {
    IDENTIFIER id-ckm-symmetric
    PARAMS TYPE ConstructionAlgorithms ARE required}

ConstructionAlgorithms ::= SEQUENCE {
    combiner                CombinerAlgorithmIdentifier,
    keyEncryptionAlgorithm  KeyEncryptionAlgorithmIdentifier
}
```

```
CombinerAlgorithmIdentifier ::=
AlgorithmIdentifier{ALGORITHM,{CombinerAlgorithms}}
CombinerAlgorithms ALGORITHM ::= {
    ... -- Expect additional objects --
}
keyTransportConstruction ALGORITHM ::= {
    IDENTIFIER id-ckm-key-transport
    PARAMS TYPE KeyEncryptionAlgorithmIdentifier ARE required
}
KeyEncryptionAlgorithmIdentifier ::=
    AlgorithmIdentifier {ALGORITHM,{KeyEncryptionAlgorithms}}
KeyEncryptionAlgorithms ALGORITHM ::= {
    ... -- Expect additional objects --
}
keyAgreeMultipleEncryptionConstruction ALGORITHM ::= {
    IDENTIFIER id-ckm-key-agree-multiple-encrypt
    PARAMS TYPE KeyEncryptionAlgorithmIdentifier ARE required
}
keyAgreeHashConstruction ALGORITHM ::= {
    IDENTIFIER id-ckm-key-agree-hash
    PARAMS TYPE DigestAlgorithmIdentifier ARE required
}
END
```

## 15 Module CMSDBKeyManagement

CMSDBKeyManagement

```
{itu-t recommendation(0) x(24) cms-profile(894) module(0) dbKeyManagement(2) version1(1)}
"/ITU-T/Recommendation/X/CMS-Profile/Module/DBKeyManagement/Version1"
```

DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

IMPORTS

```
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)}
```

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)}
```

```
Attribute{,MessageAuthenticationCodeAlgorithm
    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) }
```

```
id-dbekm-recipient-info,id-SimpleString,id-UniqueIdentifier
    FROM CMSObjectIdentifiers
    {iso(1) identified-organization(3) tc68(133) country(16) x9(840)
    x9Standards(9) x9-73(73) module(0) oids(1) v2009(1)};
```

-- X9.73 XML namespace: urn:oid:1.3.133.16.840.9.73 --

KEY-MANAGEMENT ::= TYPE-IDENTIFIER

```
DB-Encryption-Key-Management KEY-MANAGEMENT ::= {
    dbekmRecipientInfo,
    ... -- Expect additional key management objects --
```

```
    }
dbekmRecipientInfo          KEY-MANAGEMENT ::= {
    DBEKMRRecipientInfo IDENTIFIED BY id-dbekm-recv-info }
DBEKMRRecipientInfo ::= CHOICE {
    keyManager                MasterKeyEncryptedHMACKey,
    server                    DatabaseServerToKeyManager
    }
MasterKeyEncryptedHMACKey ::= SEQUENCE {
    masterKeyAID              MasterKeyAlgorithmIdentifier OPTIONAL,
    hmacKeyAID                MessageAuthenticationCodeAlgorithm OPTIONAL,
    encryptedKey              OCTET STRING(SIZE(1..MAX))
    }
MasterKeyAlgorithmIdentifier ::=
    AlgorithmIdentifier {ALGORITHM, {MasterKeyAlgorithms}}
MasterKeyAlgorithms ALGORITHM ::= {
    ... -- Expect additional algorithm objects --
    }
DatabaseServerToKeyManager ::= SEQUENCE {
    encryptedKey              MasterKeyEncryptedHMACKey,
    uniqueID                  UniqueIdentifier OPTIONAL
    -- May be known system wide --
    }
UniqueIdentifier ::= UniqueID{{SchemaIdentifier}}
SchemaIdentifier DBEKM ::= {
    simpleString,
    ... -- Expect additional schema identifier objects --
    }
simpleString DBEKM ::= {
    OID id-SimpleString PARMS SimpleString }
SimpleString ::= UTF8String(SIZE(1..MAX))
DBEKM ::= CLASS {
    &id OBJECT IDENTIFIER UNIQUE,
    &Type OPTIONAL
    }
WITH SYNTAX {OID &id [PARMS &Type]}
UniqueID{DBEKM:IOSet} ::= SEQUENCE {
    name DBEKM.&id({IOSet}),
    type DBEKM.&Type({IOSet}{@name}) OPTIONAL
    }
DbEKMAAttributeSet ::=
    SEQUENCE(SIZE(1..MAX)) OF Attribute{{DbEKMAAttributes}}
DbEKMAAttributes ATTRIBUTE ::= {
    uniqueIdentifier,
    ... -- Expect user schema identifier attributes --
    }
uniqueIdentifier ATTRIBUTE ::= {
    TYPE UniqueIdentifier IDENTIFIED BY id-UniqueIdentifier
    }
END
```

## Annex B

### Object identifiers defined in this Recommendation | International Standard

(This annex forms an integral part of this Recommendation | International Standard.)

This recommendation | International Standard defines the following object identifiers:

- a) object identifier associated to the ASN.1 signcryption module:  

```
{itu-t recommendation(0) x(24) cms-profile(894) module(0) signcryption(0)
  version1(1) }
"/ITU-T/Recommendation/X/CMS-Profile/Module/Signcryption/Version1"
```
- b) object identifier associated to the ASN.1 CKM key management module:  

```
{itu-t recommendation(0) x(24) cms-profile(894) module(0) cKMKeyManagement(1)
  version1(1) }
"/ITU-T/Recommendation/X/CMS-Profile/Module/CKMKeyManagement/Version1"
```
- c) object identifier associated to the ASN.1 DB key management module:  

```
{itu-t recommendation(0) x(24) cms-profile(894) module(0) dBKeyManagement(2)
  version1(1) }
"/ITU-T/Recommendation/X/CMS-Profile/Module/DBKeyManagement/Version1"
```
- d) object identifier associated to the signcryptedData:  

```
{itu-t recommendation(0) x(24) cms-profile(894) signcryption(1) data(0) }
"/ITU-T/Recommendation/X/CMS-Profile/Signcryption/Data"
```
- e) object identifier associated to CMS attributes  

```
{itu-t recommendation(0) x(24) cms-profile(894) signcryption(1) data(0) }
"/ITU-T/Recommendation/X/CMS-Profile/Signcryption/Data"
```
- f) object identifier associated to signerInfo attribute  

```
{itu-t recommendation(0) x(24) cms-profile(894) attribute(2) signerInfo(0) }
"/ITU-T/Recommendation/X/CMS-Profile/Attribute/SignerInfo"
```
- g) object identifier associated to signerInfos attribute  

```
{itu-t recommendation(0) x(24) cms-profile(894) attribute(2) signerInfos(1) }
"/ITU-T/Recommendation/X/CMS-Profile/Attribute/SignerInfos"
```
- h) object identifier associated to contentLocation attribute  

```
{itu-t recommendation(0) x(24) cms-profile(894) attribute(2) contentLocation(2) }
"/ITU-T/Recommendation/X/CMS-Profile/Attribute/ContentLocation"
```

## **Annex C**

### **Bibliography**

(This annex does not form an integral part of this Recommendation | International Standard.)

- [IETF RFC 5911 (2010)], New ASN.1 Modules for Cryptographic Message Syntax (CMS) and S/MIME.  
[IETF RFC 5912 (2010)] New ASN.1 Modules for the Public Key Infrastructure Using X.509 (PKIX)  
[IETF RFC 6268 (2011)] Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)
-