INTERNATIONAL STANDARD

ISO/IEC 24824-4

First edition
2021-03

Information technology — Generic applications of ASN.1 —

Part 4: Cryptographic message syntax

Reference number
ISO/IEC 24824-4:2021(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, www.iec.ch/understanding-standards.

This document was prepared by ITU-T [Telecommunication Standardization Sector of ITU] (as ITU-T X.894 [10/2018]) and drafted in accordance with its editorial rules, in collaboration with Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 6, *Telecommunications and information exchange between systems*.

A list of all parts in the ISO/IEC 24824 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

CONTENTS

INTERNATIONAL STANDARD ISO/IEC 24824-4 RECOMMENDATION ITU-T X.894

Information technology — Generic applications of ASN.1 —

Part 4: Cryptographic message syntax

1 Scope

This Recommendation | International Standard enhances the existing cryptographic message syntax (CMS) protocol by adding signcryption techniques and providing a new Abstract Syntax Notation One (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*.

2.2 Paired Recommendations | International Standards equivalent in technical content

None.

2.3 Additional References

– ISO 11568-1:2005, *Banking – Key management (retail) – Part 1: Principles*.

– ISO/IEC 11770-6:2016, *Information technology – Security techniques – Key management – Part 6: Key derivation*.

– ISO/IEC 18033-2:2006, *Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers*.

– ISO/IEC 29150:2011, *Information technology – Security techniques – Signcryption*.

– IETF RFC 5652 (2009), *Cryptographic message syntax (CMS)*.

– 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) CA certificate;

c) certificate revocation list.

The following term is defined in ISO/IEC 29150:

– signcryption

4 Abbreviations

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

ASN.1	Abstract Syntax Notation One
CEK	Content Encryption Key
CKM	Constructive Key Management
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
DBEKM	Database Encryption Key Management
DK	Data encryption Key
HK	HMAC Key
HMAC	Hashed Message Authentication
ID	Identifier
KDF	Key Derivation Function
MK	Master Key encryption key
PBKDF	Password-Based KDF
SCD	Secure Cryptographic Device
SHA	Secure Hash Algorithm
URI	Uniform Resource Identifier
XML	extensible Markup Language

5 Conventions

None.

6 Cryptographic message syntax

CMS is defined in the base text, IETF RFC 5652. ASN.1 modules have been revised to conform to the current ASN.1 standard in IETF 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 CMS 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(1)
            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 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 or attribute certificate revocation lists (CRLs) to facilitate the validation of the certificates of the signcrypters. As for **certificates**, this collection may contain more CRLs than necessary or fewer and, in the latter case, recipients have to use other means to get other CRLs.

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) **signcryptedAttributes**: content and attributes of any type or format are signcrypted.

b) **signcryptedComponents**: specified components of content of any type or format are signcrypted, and the resulting cryptogram is signed along with a manifest of signcrypted locations and other attributes.

c) **signcryptedContent**: content of any type or format is signcrypted.

d) **signcryptedEnveloped**: 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 signcrypted 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 ::= {
            signcryptedAttributes  |
            signcryptedComponents  |
            signcryptedContent           |
```



```
            signcryptedEnveloped,
            ... -- Expect additional processing modes --
            }

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

7.2.1 The signcryptedContent mode

In the signcrypted-content mode, content of any type or format is signcrypted using the signcryption algorithm identified in the **signcryptedDataAlgorithm** component of the **signcrypters** component of type **SigncryptedData**. The message sender applies this signcryption 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 **SigncrypterIDs** and the **signcrytionValue** component of type Signcrypter contains the results of signcryption.

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

```
signcryptedContent MODE ::= { ID signcrypted-content }
```

A value of type **SigncryptedData** using signcrypted-content mode is created as follows:

a) the value of the **version** component of type **SigncryptedData** is set to 1;

b) in the **contentInformation** component of type **SigncryptedData**, set the value of mode to the signcrypted-content object identifier. the optional **content** component should not be present;

c) optionally, include values in the **certificates** component of type **SigncryptedData**;

d) optionally, include values in the **crls** component of type **SigncryptedData**;

e) for each message recipient, include a value of type **Signcrypter** in the **signcrypters** component of type **SigncryptedData** 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 **signcryptedDataAlgorithm** component of type **Signcrypter** to identify the signcryption algorithm and any associated parameters,

   iv) include the results of **signcrypting** content of any type or format encoded as a value of type OCTET STRING in the **signcryptionValue** component of type **Signcrypter**,

   v) the optional **signatureInformation** component of type **Signcrypter** should not be present;

   vi) optionally, include values in the **unsigncryptedAttributes** component of type **Signcrypter**.

To recover the plaintext from the **SigncryptedData** 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 **SigncryptedData** to locate the recipient public-private key pair in the **sids** component of type **Signcrypter**;

2) designcrypt the value in the **signcryptionValue** 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 signcryption algorithm and associated parameters provided in the **signcryptedDataAlgorithm** component of type **Signcrypter**.

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

7.2.2 The signcryptedAttributes mode

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

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

```
signcryptedAttributes MODE ::= { ID signcrypted-attributes }
```

A value of type **SigncryptedData** using signcrypted-attributes mode is created as follows:

a) the value of the **version** component of type **SigncryptedData** is set to 1;

b) in the **contentInformation** component of type **SigncryptedData**, set the value of mode to the **signcrypted-attributes** object identifier. the optional **content** component should not be present;

c) optionally, include values in the **certificates** component of type **SigncryptedData**;

d) optionally, include values in the **crls** component of type **SigncryptedData**;

e) for each message recipient, include a value of type **Signcrypter** in the **signcrypters** component of type **SigncryptedData** 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 indentify the public-private key pairs of the message sender and recipient;

   iii) set the value of the **signcryptedDataAlgorithm** component of type **Signcrypter** to identify the signcryption algorithm and any associated parameters;

   iv) create a value of type **ToBeSigncrypted** by setting its **content** component to a value of type **Content** and its **attributes** component to a value of type **SigncryptedAttributes**;

   v) signcrypt an encoded value of type **ToBeSigncrypted** and include the results of the processing in the **signcryptionValue** component of type **Signcrypter**;

   vi) the optional **signatureInformation** component of type **SigncrypterInfo** should not be present;

   vii) optionally, include values in the **unsigncryptedAttributes** component of type **Signcrypter**.

To recover the plaintext from the **SigncryptedData** 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 **SigncryptedData** to locate the recipient public-private key pair in the **sids** component of type **Signcrypter**;

b) designcrypt the value in the **signcryptionValue** component of type **Signcrypter** for the recipient using the signcryption algorithm and any associated parameters provided in the **signcryptedDataAlgorithm** component of type **Signcrypter** to recover a value of type **ToBeSigncrypted**;

c) perform certificate path validation to gain assurance that the sender public key certificate is trusted.

7.2.3 The signcryptedComponents mode

In signcrypted-components mode, 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. A content-type specific manifest of signcrypted element locations in the content is bound to the partially signcrypted content under a digital signature.

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

```
signcryptedComponents MODE ::= { ID signcrypted-components }
```

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

```
SigncryptedPartsManifest ::= Signcrypted{{Manifest}}

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

SIGNCRYPTED ::= CLASS {
```



```
 &id    OBJECT IDENTIFIER UNIQUE,
 &Type  OPTIONAL
}
WITH SYNTAX {OID &id [PARMS &Type]}

Manifest SIGNCRYPTED ::= {
              xPathManifest,
               ...
              }
```

The type of manifest required to identify the signcrypted components of a particular object varies with the type, structure and format of the object. For content in the form of an image, a **signcrypted** 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 signcrypted and thereby redacted.

In this Recommendation | International Standard, a single manifest object is defined to support component identification in an extensible markup language (XML) document. The **xPathManifest** object uses location paths based on the XPath query language to specifiy a set of one or more signcrypted XML document components and is defined as follows:

```
xPathManifest SIGNCRYPTED ::= {
              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 **SigncryptedData** using signcrypted-components mode is created as follows:

a) the value of the **version** component of type **SigncryptedData** is set to 1;

b) in the **contentInformation** component of type **SigncryptedData**, set the value of **mode** to the signcrypted-components object identifier. the optional **content** component should not be present;

c) optionally, include values in the **certificates** component of type **SigncryptedData**;

d) optionally, include values in the **crls** component of type **SigncryptedData**;

e) for each message recipient, include a value of type **Signcrypter** in the **signcrypters** component of type **SigncryptedData** 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 signcrypted results as the value of the **signcryptionValue** 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 ID, identify the signature algorithm object identifier and any associated parameters in the **signatureAlgorithm** component of type **Signcrypter**,

   vi) prepare a manifest value in the **signcryptedPartsManifest** component of type **ToBeSigned** that indicates the location of each signcrypted element in the **signcryptionValue** 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 **unsigncryptedAttributes** component of type **Signcrypter**.

To recover the plaintext from the **SigncryptedData** 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 **SigncryptedData** 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 signcryption algorithm and any associated parameters provided in the **signcryptedDataAlgorithm** component of type **Signcrypter**;

c) using the manifest in the **signcryptedPartsManifest** component of type **ToBeSigned**, designcrypt each of the indicated signcrypted elements listed in the manifest to recover the partially signcrypted components of the content;

d) perform certificate path validation to gain assurance that the sender certificates are trusted.

7.2.4 The signcryptedEnvelope mode

In the signcrypted-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 ANSI X9.73-2017 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 **SigncryptedData**. The **contentType** component of type **EncryptedContentInfo** is set to the information object identifier value signcrypted-envelope.

The **SigncryptedData** message contains one value of type **Signcrypter** for each message recipient. The **Signcrypter** type contains a **signcryptionAttributes** component that must contain the symmetric encryption key signcrypted for each message recipient in a **signcryptedEnvelope** attribute.

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

```
signcryptedEnvelope ATTRIBUTE ::= {
            WITH SYNTAX SigncryptedKey ID signcrypted-envelope
            }

SigncryptedKey ::= OCTET STRING
```

When the **signcryptedEnvelope** attribute is included in the **signcryptionAttributes** component of type **Signcrypter**, both the **messageDigest** and **contentType** attributes shall be present. To complete construction of the message, the **eContent** is concatenated together with all **signcryptionAttributes** 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 **signcryptedEnvelope** attribute, a message recipient must first designcrypt 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 **SigncryptedData** using signcrypted-envelope mode is created as follows:

a) the value of the **version** component of type **SigncryptedData** is set to 1;

b) in the **contentInformation** component of type **SigncryptedData**, set the value of mode to the signcrypted-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 **SigncryptedData**;

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 **ToBeSigncrypted** that includes this **messageDigest** attribute in the **attributes** component, and the symmetric content encryption key (CEK) in the **contents** component, a value of type **ToBeSigncrypted** that will be signcrypted for each message recipient;

h) optionally, include values in the **certificates** component of type **SigncryptedData**;

i) optionally, include values in the **crls** component of type **SigncryptedData**;

j) for each message recipient, include a value of type **Signcrypter** in the **signcrypters** component of type **SigncryptedData** 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 **signcryptedDataAlgorithm** component of type **Signcrypter** to identify the signcryption algorithm and any associated parameters,

   iv) include the results of signcrypting the prepared value of type **ToBeSigncrypted** in the **signcryptionValue** component of type **Signcrypter**,

   v) the optional **signatureInformation** component of type **Signcrypter** should not be present,

   vi) Optionally, include values in the **unsigncryptionAttributes** component of type **Signcrypter**.

To recover the plaintext from the **SigncryptedData** 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 **SigncryptedData** to locate the recipient public-private key pair in the **sids** component of type **Signcrypter**;

b) decode the optional **content** component in the **contentInformation** component of type **SigncryptedData**, to recover a value of type **NamedKeyEncryptedData** from the octet string;

c) designcrypt the value in the **signcryptionValue** 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 signcrypted value of type **ToBeSigncrypted** that contains a symmetric key and a message digest of the content encypted 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 signcrypted value of type **ToBeSigncrypted** and comparing the two digests for equivalency;

f) use the symmetric key recovered from designcrypting the value of type **ToBeSigncrypted** to decrypt the content value of the **contentInformation** component of type **SigncryptedData**.

7.3 The Signcrypter type

```
Signcrypter  ::= SEQUENCE {
             Version                Version,
             Sids                   SigncrypterIDs,
             signcryptedDataAlgorithm     SigncryptedDataAlgorithmIdentifier,
             signcryptionValue            SigncryptionValue,
             signatureInformation   SignatureInformation  OPTIONAL,
             unsigncryptedAttributes      UnSigncryptedAttributes  OPTIONAL
             }
```

The public-private key pairs of the message sender and recipient are identified as they are in the **SigncryptedData** 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 signcrypted by the sender.

The **SigncryptedDataAlgorithmIdentifier** type contains the signcryption algorithm ID and any associated parameters. For all of the signcryption algorithms specified in ISO/IEC 29150, the parameters include one of the hash functions and one of the key derivation functions (KDFs) specified in ISO/IEC 18033-2.

```
SigncryptedDataAlgorithmIdentifier ::=
             AlgorithmIdentifier {{SigncryptAlgorithms}}

SigncryptAlgorithms ALGORITHM ::= {
             SigncryptionMechanism,  -- ISO/IEC 29150 Signcryption algorithms --
              ...  -- Expect additional algorithm objects --
             }
```

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

```
SigncryptionValue ::= 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 signcrypted-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 {
             signcryptedPartsManifest     SigncryptedPartsManifest,
             signedAttributes       SignedAttributes
             }

SigncryptedPartsManifest ::= Signcrypted{{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 **unsigncryptedAttributes** 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 signcrypted data processing. Type **UnSigncryptedAttributes** is defined as follows:

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

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

8 Quantum safe SignedData signatures

Anticipated attacks on current digital signature algorithms using quantum computers will subject documents signed today and requiring long-term protection to increasing security risk over time. This problem is not limited solely to quantum computing risks, such as risk of repudiation by the signer. Long-term signed documents such as a 30 year mortgage are subject to similar risks.

These risks arise as new attacks on signature algorithms are discovered and key length requirements grow with computing power advances. An increase in the number of attacks, the continuing rise in legal and regulatory risks, and changes to the security polices of organizations all add to these risks. It is possible to mitigate some of these security risks using a quantum-safe countersignature over signed content created using current digital signature algorithms.

A countersignature that relies on a quantum-safe signature can be implemented in a SignedData message using an optional signed attribute. The schema of a **SignedData** message supports a series of signers represented in a value of type **SignerInfos**. Each signer is represented in this series as a value of type **SignerInfo**.

Type **SignerInfo** is defined as:

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

Type **SignerInfo** allows each signer to use a different signing key, message digest, and signature algorithm. Each signer can include their own set of attributes that will be cryptographically bound under their signature. A signerInfos attribute collects this series of values into an attribute that can be included in the signed attributes of a counter-signer.

The **signerInfos** attribute is as follows:

```
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)}
```

The **signerInfos** attribute contains a value of type **SignerInfos**, a series of values of type **SignerInfo**, one value for each signer of the **SignedData** content. Each cosigner shall sign content using their choice of signature and message digest algorithm. The signing key of each cosigner shall be included in the **SignerInfo** value of the cosigner using any of the choice alternatives defined in the **SignerIdentifier** type.

The optional **signedAttrs** component of type **SignerInfo** shall be present in the message. At a minimum, each cosigner shall include a **contentType** attribute and a **messageDigest** attribute in the **signedAttrs** component of their **SignerInfo** value. Additional signed attributes of any type or format may also be included by each cosigner. Any number or type of unsigned attributes may also be included by each cosigner in the **unsignedAttrs** component of their **SignerInfo** value.

The values of type **SignerInfo** created by all signers shall follow other defined processing. During signature verification of a **SignedData** message, a relying party may treat the **signerInfos** attribute as an opaque string. Applications that recognize this attribute may choose to defer signature verification processing. Failure of one or more cosigner **SignerInfo** values shall be handled as defined by the application.

8.1 Detached content consideration

When the detached form of **SignedData** is used, the content of the message is not present in the **SignedData** type. This message content must be available during signing and signature verification operations so that a message digest of the signed content can be calculated.

When the default content location is not known to the communicating parties, content signers can include a **contentLocation** attribute in their signed attributes. This attribute can also be used when it is necessary for a consigner to indicate a different detached object, such as a language-specific version of a contract.

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

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

URI ::= UTF8String(SIZE(1..MAX))

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

A value of type **URI** is a uniform resource identifier (URI) value that points to a location of detached **SignedData** content. A **contentLocation** attribute can be included in a **SignedAttributes** component of a **SignerInfo** component of type **SignedData** of any signer of the message. In some applications, it may be convenient to include a single content location attribute in the signed attributes of the countersigner.

8.2 Time stamp consideration

Time stamps included in **SignedData** can be used to demonstrate that the validity period of a signer certificate included the time of signing a message. Long-term signatures may need to be verified after the validity period of a signing certificate has expired. A time stamp attribute that is included in the **SignedAttributes** component can be compared by a relying party to the validity period of the signer certificate to ensure the certificate was valid for use when the message was signed.

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

```
aa-timeStamped ATTRIBUTE ::=
            {TYPE TimeStamped IDENTIFIED BY id-timeStamped}

TimeStamped  ::= SEQUENCE {
            timeStampValue   TimeStamp,
            timeStampService URI OPTIONAL
            }

TimeStamp ::= CHOICE {
            timeStampToken   TimeStampToken,
            localTimeStamp   GeneralizedTime,
            ... -- Expect additional time types --
            }

id-timeStamped OBJECT IDENTIFIER ::= {itu-t recommendation(0) x(24)
            cms-profile(894) attribute(2) timeStamped(5)}
```

The type **TimeStamped** contains two components, a required **timeStampValue** and an optional **timeStampService** that indicates the location of a time stamp service provider that can validate the time stamp. A **timeStampValue** component is a value of type **TimeStamp**, a choice between two alternatives, an ANSI X9.95 trusted timestamp token or a value from a local time source.

```
TimeStamp ::= CHOICE {
            TimeStampToken    TimeStampToken, -- X9.95 Trusted Time Stamp --
            localTimeStamp    GeneralizedTime
            }
```

The first choice alternative of type **TimeStamp** may be any of the four types of tokens defined in ANSI X9.95.

8.3 The tokenizedParts attribute

The **tokenizedParts** attribute carries a manifest, a list of the locations of elements within content of any type or format that has been tokenized. Tokenization techniques provide confidentiality services for the elements of the content specified in the manifest. A manifest can be constructed on a per-message recipient basis to achieve granular message redaction. The format and the information contained in the manifest varies with the type of content and user requirements. Only one manifest is defined in this Recommendation | International Standard, a simple manifest for XML content, though the set of possible manifests is extensible and unbounded.

XML Path (XPath) expressions can be used to locate any tokenized element in any XML-instance document. To identify the tokenization components in an XML-instance document, a set of XPath expressions can be used to identify the location of each tokenized element. When the contents of an XML element are tokenized, the sender includes the outer markup tags in the tokenization operation.

These outer tags are not removed from the document, since they are used to locate the element using XPath and compared for assurance to the bounding tags in a detokenized value. The markup between these outer tags is replaced with a character string representation of the tokenization results, a value of XML type base64Binary. This XML type is used to represent arbitrary Base64-encoded binary data in a human-readable, 'mail-safe' format. A message recipient uses the list of XPath expressions to locate the tags in an XML-instance document that contain tokenized data.

The entire document containing tokenized components and the associated manifest attribute are signed to cryptographically bind them under a digital signature. The signature prevents any part of the document or attributes from being substituted or modified without detection.

Successful signature verification provides a relying party assurance of the authenticity of the document and the manifest attribute. A tokenization manifest attribute is protected under the signature and indicates that the tokenization components of the document have integrity and confidentiality. The signature on the partially tokenized content can then be verified and its plaintext content recovered. The recovered plaintext can then be used to replace the tokenized value and surrounding tags with the recovered XML markup prior to performing XML schema validation

9 Other key management techniques

9.1 Constructive key management

Constructive key management (CKM) is a dynamic symmetric key management framework described in ANSI X9.69-2017. 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 using a mathematical function to produce an object key (content key).

Internal keys associated with CKM meet criteria established in ISO 11568-1. The mathematical function inherent in CKM is a two-part process for binding an analogue 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 that includes a random number and other symmetric key functionality, which together results in a CEK. 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 CEK. 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 ANSI X9.69-2017.

```
ckmRecipientInfo KEY-MANAGEMENT ::=
            { KeyConstructRecipientInfo  IDENTIFIED BY id-ckm-recip-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-recip-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-recip-info -->
  </oriType>
  <oriValue>
    <KeyConstructRecipientInfo>
```

```
        <version> 1 </version>
        <did>
           <domainName>domainName</domainName>
           <domainMaintenanceLevel>1</domainMaintenanceLevel>
        </did>
        <ckmid>
           <unencrypted> 1 2 3</unenrypted>
        </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.

9.2 Database encryption key management

Database encryption key management (DBEKM) is a key management technique defined in ANSI 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-recip-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-recip-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 (DK) in cryptographic software and using a master key in cryptographic hardware, also referred to as an 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 (PBKDF) to protect data encryption keys.

The DBEKM method employs three key types:

1) master key encryption key (MK) used to protect the hashed message authentication (HMAC) key (HK);
2) keyed HK used to generate a seed;
3) 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 HMAC;
3) algorithms for hashing, e.g. secure hash algorithm 2 (SHA-2), secure hash algorithm 3 (SHA-3);
4) algorithms for KDFs, i.e. ISO/IEC 11770-6.

The DBEKM method has the following benefits:

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

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 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 DKs with a single server when protecting different data elements, and using multiple DKs or MKs with multiple servers.

9.2.1 Single server initial data encryption key

For a single server initial DK, 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 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 SDC 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.

9.2.2 Single server data encryption key change

For a single server DK 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.

9.2.3 Multiple data encryption keys with a 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 managed 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 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 SDC decrypts the HK using the MK, generates seed s1 using the HMAC algorithm with the HK and the unique ID1, and sends seed s1 to the server over a secured channel. The server generates the first DK1 using a KDF with seed s1 and installs the DK1 into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and 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 SDC decrypts the HK using the MK, generates seed s2 using the HMAC algorithm with the HK and the unique ID2, and sends seed s2 to the server over a secured channel. The server generates the second DK2 using a KDF with seed s2 and installs the DK2 into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and 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.

9.2.4 Multiple data encryption keys with multiple servers

Multiple servers collocated within the same data-centre 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 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 SDC decrypts the HK using the MK, generates seed s1 using the HMAC algorithm with the HK and the unique ID1, and sends seed s1 to server S1 over a secured channel. The server S1 generates DK1 using a KDF with seed s1 and installs the DK1 into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and 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 seed s2 to server S2 over a secured channel. The server generates DK2 using a KDF with seed s2 and installs the DK2 into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and 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.

9.2.5 Multiple hashed message authentication keys with multiple servers

Multiple servers deployed regionally in different data-centres might need a unique HK per server instead of sharing the same key on all servers. The SCD generates a single MK and multiple HKs, one for 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 one 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 seed s1 to server S1 over a secured channel. The server S1 generates DK1 using a KDF with seed s1 and installs the DK1 into memory for data encryption and decryption. Meanwhile, the SCD destroys HK1 and 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 seed s2 to server S2 over a secured channel. The server generates DK2 using a KDF with seed s2 and installs the DK2 into memory for data encryption and decryption. Meanwhile, the SCD destroys the HK and seed s2. When any of the servers is 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.)

A.1 Main CMS module (from IETF 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) identified-organization(3) dod(6) internet(1) security(5)
               mechanisms(5) pkix(7) id-mod(0) id-mod-v1AttrCert-02(49)}

CMSProfileAttributes
               FROM CMSProfileAttributes
               {itu-t recommendation(0) x(24) cms-profile(894) module(0)
               cMSProfileAttributes(3) version1(1)}

tokenizedParts
               FROM TokenizationManifest
               {iso(1) identified-organization(3) tc68(133) country(16)
               x9(840) x9Standards(9) x9-73(73) module(0) tokeMan(7)};

-- 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,
                encapContentInfo  EncapsulatedContentInfo,
                certificates      [0] IMPLICIT CertificateSet OPTIONAL,
                crls              [1] IMPLICIT RevocationInfoChoices OPTIONAL,
                signerInfos       SignerInfos }

SignerInfos ::= SET OF SignerInfo

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

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

SignedAttributes ::= Attributes {{ SignedAttributesSet }}

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

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

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-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
```

A.2 Module CMSObjectIdentifiers

```
CMSObjectIdentifiers
               {iso(1) identified-organization(3) tc68(133) country(16) x9(840)
               x9Standards(9) x9-73(73) module(0) oids(1) v2009(1)}

DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- EXPORTS All --

-- IMPORTS All --

OID ::= OBJECT IDENTIFIER -- Alias

-- Content types, from RSA PKCS #7 and IETF S/MIME

pkcs7 OID ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) }

id-data OID ::= { pkcs7 data(1) }

id-signedData OID ::= { pkcs7 signedData(2) }

id-envelopedData OID ::= { pkcs7 envelopedData (3) }

id-digestedData OID ::= { pkcs7 digestedData(5) }

id-encryptedData OID ::= { pkcs7 encryptedData (6) }

id-namedkeyencryptedData OID ::= { iso(1) member-body(2) us(840)
               x973(10060) types(1) namedKeyEncryptedData(2) }

-- Signcryption object identifiers --

id-signcryptedData OID ::= itu-t recommendation(0) x(24) cms-profile(894) signcryption(1)
data(0)}

-- The signcryption-manifest arc is the root identifier for all --

-- SigncryptedData manifest types definied in this standard. --

signcryption-manifest OID ::= { id-signcryptedData manifest(1) }

-- A value of xPath identifies the XPathSet manifest type. --

xPath OID ::= { signcryption-manifest xPath(0) }

pkcs9 OID ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) }

smime OID ::= { pkcs9 smime(16) }

id-ct-authData OID ::= { smime ct(1) 2 }
```

```
-- Signed attributes, from RSA PKCS #9, IETF S/MIME, and X9.73 --

id-contentType OID ::= { pkcs9 contentType(3) }

id-messageDigest OID ::= { pkcs9 messageDigest(4) }

-- Authenticated attribute, from IETF S/MIME --

-- CKM key management object identifiers --

id-ckm-recip-info OID ::= { iso member-body(2) us(840) x973(10060) km(2) 1 }

id-ckm-recip-info2 OID ::= { iso member-body(2) us(840) x973(10060) km(2) 2}

id-ckm-algorithms OID ::= { iso member-body(2) us(840) x973(10060) algorithms(3) }

id-ckm-symmetric OID ::= { id-ckm-algorithms symmetric(1) }

id-ckm-key-transport OID ::= { id-ckm-algorithms key-transport(2) }

id-ckm-key-agree-multiple-encrypt OID ::= {

id-ckm-algorithms key-agree-multiple-encrypt(3) }

id-ckm-key-agree-hash OID ::= { id-ckm-algorithms key-agree-hash(4) }

id-ckm-header OID ::= { iso member-body(2) us(840) x973(10060) header(4) }
              ckm-CMS OID ::= {
              joint-iso-itu-t(2)   international-organizations(23)   set(42)   vendors(9)
              griffin(10) business(3) tecsec(0) cms(2) header(2) }

id-Ivec OID ::= { ckm-CMS 1 }

id-Secryptm OID ::= { ckm-CMS 2 }

id-Filelength OID ::= { ckm-CMS 3 }

id-Filehash OID ::= { ckm-CMS 4 }

id-Filename OID ::= { ckm-CMS 5 }

id-Domainlist OID ::= { ckm-CMS 6 }

id-Accessgrouplist OID ::= { ckm-CMS 7 }

id-Issuer OID ::= { ckm-CMS 8 }

id-Credentiallist OID ::= { ckm-CMS 9 }

id-SignKey OID ::= { ckm-CMS 10 }

id-KeyUsage OID ::= { ckm-CMS 11 }

id-BitSpray OID ::= { ckm-CMS 12 }

id-BitSprayMeta OID ::= { ckm-CMS 12 1 }

id-BitSprayShares OID ::= { ckm-CMS 12 2 }

id-FavoriteName OID ::= { ckm-CMS 13 }

id-DataSignature OID ::= { ckm-CMS 14 }

id-BlockSize OID ::= { ckm-CMS 15 }

id-DataFormat OID ::= { ckm-CMS 16 }

-- Tokenization object identifiers --

id-tokenization-manifest OID ::= { iso(1) identified-organization(3) tc68(133)
              country(16) x9(840) x9Standards(9) x9-73(73) tokenization(3) }

id-tokenizedParts OID ::= { id-tokenization-manifest tokenizedParts(0) }

id-XPathTokensSet OID ::= { id-tokenization-manifest xPathTokensSet(1) }

-- Signcryption processing mode object identifiers --

signcryption OID ::= { iso(1) identified-organization(3) tc68(133)

country(16) x9(840) x9Standards(9) x9-73(73) signcryption(4) }

signcryption-mode OID ::= { signcryption modes(1) }
```



```
signcrypted-content OID ::= { signcryption-mode content(1) }

signcrypted-attributes OID ::= { signcryption-mode attributes(2) }

signcrypted-components OID ::= { signcryption-mode components(3) }

signcrypted-envelope OID ::= { signcryption-mode enveloped(4) }

-- Signcryption object identifiers --

id-signcryptedParts OID ::= { signcryption-manifest signcryptedParts(1) }

id-XPathSigncryptionSet OID ::= { signcryption-manifest xPathSigncryptionSet(2) }

-- X9.73 attribute object identifiers --

id-cms-attributes OID ::= { iso(1) identified-organization(3) tc68(133)

country(16) x9(840) x9Standards(9) x9-73(73) attributes(1) }

xmlMarkup OID ::= { id-cms-attributes xml(0) }

id-cms-SAML OID ::= { xmlMarkup saml(1) }

id-cms-XKMS OID ::= { xmlMarkup xkms(2) }

id-messageComponents OID ::= { xmlMarkup mc(3) }

-- X9.73 XML namespace prefix values --

id-cms-namespaces OID ::= { iso(1) identified-organization(3) tc68(133)

country(16) x9(840) x9Standards(9) x9-73(73) namespaces(2) }

cms OID ::= { iso(1) identified-organization(3) tc68(133) country(16) x9(840)
              x9Standards(9) x9-73(73) namespaces(2) cms(0) }

-- RFC 3211 password-based encryption --

id-alg-PWRI-KEK OBJECT IDENTIFIER ::= { iso(1) member-body(2)

us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) alg(3) 9 }

-- Database Bncryption Key Management --

dbEKM OID ::= {joint-iso-itu-t(2) country(16) us(840) organization(1) wfbna(114171)
              lobs(4) eisArchitecture(1) techniques(2) dbEKM(0)}

id-SimpleString OID ::= { dbEKM ss(1) }

id-UniqueIdentifier OID ::= { dbEKM uid(2) }

id-dbekm-recip-info OID ::= { iso member-body(2) us(840) x973(10060) km(2) 3 }

END -- CMSObjectIdentifiers --
```

A.3 Module AlgorithmInformation-2009 (from IETF 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 sufficent 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 (ITU-T 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
```

A.4 Module CryptographicMessageSyntaxAlgorithms-2009 (from IETF 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 ::= dhpublicnumber

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

kwa-3DESWrap KEY-WRAP ::= {
```



```
                IDENTIFIER id-alg-CMS3DESwrap
                PARAMS TYPE NULL ARE required
                SMIME-CAPS {IDENTIFIED BY id-alg-CMS3DESwrap}
                }

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

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

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

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

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

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

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

END
```

A.5 Module PKIX-Algs-2009 (from IETF 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
```

A.6 Module PKIXAttributeCertificate-2009 (from IETF 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 IETF RFC 3281

-- 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 {
                notBeforeTim       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
                }

AAControls ::= 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
                   }
               }

SvceAuthInfo ::=    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
```

A.7 Module AttributeCertificateVersion1-2009 (from IETF RFC 5912)

```
AttributeCertificateVersion1-2009
```



```
                {iso(1)    identified-organization(3)    dod(6)    internet(1)    security(5)
                mechanisms(5) pkix(7) id-mod(0) id-mod-v1AttrCert-02(49)}

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 ITU-T X.509-1997, 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
```

A.8 Module PKIX-CommonTypes-2009 (from IETF 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
```

A.9 Module PKIX-X400Address-2009 (from IETF 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
```

A.10 Module PKIX1Explicit-2009 (from IETF 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 IETF 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
```

A.11 Module PKIXImplicit-2009 (from IETF 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 ITU-T 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
```

A.12 Module PKIX1-PSS-OAEP-Algorithms-2009 (from IETF 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
```

A.13 Module SecureMimeMessageV3dot1-2009 (from IETF 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-pkalgs-02(54)};

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

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

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

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

aa-smimeCapabilities ATTRIBUTE ::= {
               TYPE SMIMECapabilities{{SMimeCapsSet}}
               IDENTIFIED BY smimeCapabilities }

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

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

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

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

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

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

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

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

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



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

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

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

--  The following list OIDs to be used with S/MIME V3
-- Signature Algorithms Not Found in [RFC3370]
--
-- md2WithRSAEncryption OBJECT IDENTIFIER ::=
--    {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-1(1)
--     2}
--
-- Other Signed Attributes
--
-- signingTime OBJECT IDENTIFIER ::=
--    {iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
--     5}
--    See [RFC5652] for a description of how to encode the attribute
--    value.

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

SMIMECapabilitiesParametersForRC2CBC ::= INTEGER (40 | 128, ...)

--    (RC2 Key Length (number of bits))

END
```

A.14 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,
               signcryptedDataAlgorithm  SigncryptedDataAlgorithmIdentifier,
               signcryptionValue         SigncryptionValue,
               signatureInformation      SignatureInformation OPTIONAL,
               unsigncryptedAttributes   UnSigncryptedAttributes OPTIONAL
               }

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

KeyPairIdentifier ::= SignerIdentifier

ToBeSigncrypted ::= SEQUENCE {
               content     Content,
               attributes  SigncryptedAttributes
               }

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

SigncryptionAttributes ATTRIBUTE ::= {
               SignedAttributesSet       | -- CMS Signed Attributes --
```



```
                signcryptedEnvelope,
                ... -- Expect user defined attributes --
                }

signcryptedEnvelope ATTRIBUTE ::= {
                TYPE SigncryptedKey IDENTIFIED BY signcrypted-envelope
                }

SigncryptedKey ::= OCTET STRING

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

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

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

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

ToBeSigned ::= SEQUENCE {
                signcryptedPartsManifest SigncryptedPartsManifest,
                signedAttributes          SignedAttributes
                }

SigncryptedPartsManifest ::= Signcrypted{{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 --})

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

A.15 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-recip-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}

ECDomainParameters
                FROM ANSI-X9-62
                {iso(1) member-body(2) us(840) 10045 modules(0) 2};

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

ckmRecipientInfo EXTENDED-KEY-MGMT-INFO ::= {
                KeyConstructRecipientInfo IDENTIFIED BY id-ckm-recip-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 X9.62 --
                }
```



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

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
```

A.16 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-recip-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 ::= {
               DBEKMRecipientInfo IDENTIFIED BY id-dbekm-recip-info }

DBEKMRecipientInfo::= 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
               }

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

DbEKMAttributes ATTRIBUTE ::= {
               uniqueIdentifier,
               ... -- Expect user schema identifier attributes --
               }

uniqueIdentifier ATTRIBUTE ::= {
               TYPE UniqueIdentifier IDENTIFIED BY id-UniqueIdentifier
               }

END
```

A.17 Module CMSProfileAttributes

```
CMSProfileAttributes

{itu-t recommendation(0) x(24) cms-profile(894) module(0)

       cMSProfileAttributes(3) version1(1)}

"/ITU-T/Recommendation/X/CMS-Profile/Module/CMSProfileAttributes/Version1"

DEFINITIONS AUTOMATIC TAGS ::=

BEGIN

IMPORTS

ATTRIBUTE
               FROM PKIC-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)}

SignerInfo,SignerInfos,DigestedData
               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) }

TimeStampToken
               FROM TrustedTimeStamp
               {   iso(1)   identified-organization(3)   tc68(133)   country(16)   x9   (840)
               x9standards(9)
                   x9-95(95) module(0) tts(1) v2010-2016(1)};

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(SIZE(1..MAX))

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

aa-contentLocations ATTRIBUTE ::=
               {TYPE URIs IDENTIFIED BY id-contentLocations}

URIs ::= SEQUENCE SIZE(1..MAX) OF uri URI

id-contentLocations OBJECT IDENTIFIER ::= {itu-t recommendation(0) x(24)
               cms-profile(894) attribute(2) contentLocations(3)}

aa-precedingBlock ATTRIBUTE ::=
               {TYPE HashPointer IDENTIFIED BY id-precedingBlock}

HashPointer    ::= SEQUENCE {
               hash        DigestedData OPTIONAL,
               pointers    Pointers     OPTIONAL
               } ((WITH COMPONENTS {...,hash PRESENT}) |
                  (WITH COMPONENTS {...,pointers PRESENT}))

Pointers ::= SEQUENCE SIZE(1..MAX) OF pointer Pointer

Pointer ::= CHOICE {
               uri         URI,          -- Uniform Resource Identifier
               rfid        RFID,         -- Radio Frequency Identification
               gps         GPS,          -- Global Positioning System
               address     Address,      -- Free format object location
               dbRecord    DBRecord,     -- Number of fully qualified name
               ...         -- Expect other pointer types
               }

RFID ::= OCTET STRING

GPS  ::= OCTET STRING

Address ::= UTF8String

DBRecord   ::= UTF8String

id-precedingBlock OBJECT IDENTIFIER ::= {itu-t recommendation(0) x(24)
               cms-profile(894) attribute(2) precedingBlock(4)}

aa-timeStamped ATTRIBUTE ::=
               {TYPE TimeStamped IDENTIFIED BY id-timeStamped}

TimeStamped    ::= SEQUENCE {
               timeStampValue     TimeStamp,
               timeStampService   URI OPTIONAL
               }

TimeStamp  ::= CHOICE {
               timeStampToken     TimeStampToken,
               localTimeStamp     GeneralizedTime,
               ... -- Expect additional time types --
               }

id-timeStamped OBJECT IDENTIFIER ::= {itu-t recommendation(0) x(24)
               cms-profile(894) attribute(2) timeStamped(5)}

aa-sidechains  ATTRIBUTE ::=
               {TYPE Sidechains   IDENTIFIED BY id-sidechains}

Sidechains ::= SEQUENCE (SIZE(0..MAX)) OF linked Sidechain

Sidechain  ::= HashPointer

id-sidechains OBJECT IDENTIFIER ::= {itu-t recommendation(0) x(24)
               cms-profile(894) attribute(2) sidechains(6)}

aa-parentBlock ATTRIBUTE ::=
               {TYPE ParentBlock IDENTIFIED BY id-parentBlock}

ParentBlock::= HashPointer

id-parentBlock OBJECT IDENTIFIER ::= {itu-t recommendation(0) x(24)
               cms-profile(894) attribute(2) parentBlock(7)}
```



```
CMSProfileAttributes ATTRIBUTE ::= {
               aa-signerInfo | aa-signerInfos | aa-contentLocation |
               aa-contentLocations | aa-precedingBlock | aa-timeStamped |
               aa-sidechains | aa-parentBlock, ...}

END
```

A.18 Module TokenizationManifest

```
TokenizationManifest {
               iso(1) identified-organization(3) tc68(133) country(16)
               x9(840) x9Standards(9) x9-73(73) module(0) tokeMan(7) }

DEFINITIONS AUTOMATIC TAGS ::= BEGIN

-- EXPORTS All --

IMPORTS

-- X9.73 Cryptographic Message Syntax (CMS) --

ATTRIBUTE

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)}

-- X9.73 CMS Object Identifiers --

id-tokenizedParts, id-XPathTokensSet
               FROM CMSObjectIdentifiers {
               iso(1) identified-organization(3) tc68(133) country(16) x9(840)
               x9Standards(9) x9-73(73) module(0) oids(1) v2009(1)};

TokenizedParts ::= Tokenized {{ Manifest }}

Manifest TOKENIZED ::= {
               xPathTokensManifest,
               ...  -- Expect additional manifest objects --
               }

xPathTokensManifest TOKENIZED ::= {
               OID id-XPathTokensSet PARMS XPathTokensSet
               }

XPathTokensSet ::= SEQUENCE {
               tSP       TokenServiceProvider OPTIONAL,
               xPathSet  XPathSet
               }

TokenServiceProvider ::= URI

URI ::= UTF8String (SIZE(1..MAX))

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

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

tokenizedParts ATTRIBUTE ::= {
               TYPE TokenizedParts IDENTIFIED BY id-tokenizedParts
               }

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

Tokenized { TOKENIZED:IOSet } ::= SEQUENCE {
               name  TOKENIZED.&id({IOSet}),
               parts TOKENIZED.&Type({IOSet}{@name}) OPTIONAL
               }

END  -- TokenizationManifest –
```

A.19 Module TransientKey

```
TransientKey {
               iso(1) identified-organization(3) tc68(133) country(16) x9(840)
               x9Standards(9) x9-95(95) module(0) tk(2) version(0) v2010-2016(1)}
DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS All; --

IMPORTS

-- X9.73 Cryptographic Message Syntax (CMS) ASN.1 and XML --

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


-- X9.95 Trusted Time Stamp --

ALGORITHM, OID, TSTInfo
               FROM TrustedTimeStamp {
               iso(1) identified-organization(3) tc68(133) country(16) x9(840)
               x9Standards(9) x9-95(95) module(0) tts(1) v2010-2016(1) }
Certificate
               FROM AuthenticationFramework {
               joint-iso-itu-t ds(5) module(1) authenticationFramework(7) 8};

transientKeySignedTST OID ::= {
               iso(1) identified-organization(3) tc68(133) country(16) x9(840)
               x9Standards(9) x9-95(95) module(0) tk(2) contentType(1)}

tsp-req-tk OID ::= {
               iso(1) identified-organization(3) tc68(133) country(16) x9(840)
               x9Standards(9) x9-95(95) module(0) tk(2) method(2)}

TransientKeySignedTST ::= SEQUENCE {
               version         TKSVersion,
               tstAndInterval  TSTAndInterval,
               digest          Digest,
               previousDigest  [0] Digest  OPTIONAL,
               signature       Signature

}

TKSVersion ::= INTEGER { version1(1) } (version1, ...)

TSTAndInterval ::= SEQUENCE {
               tstInfo       TSTInfo,
               intervalInfo  IntervalInfo
               }

IntervalInfo ::= SEQUENCE {
               version             IIVersion,
               signedIntervalSpec  SignedIntervalSpec,
               archiveTree         ArchiveTree,
               certifierList       UriList  OPTIONAL
               }

IIVersion ::= INTEGER { version1(1) } (version1, ...)

SignedIntervalSpec ::= SEQUENCE {
               intervalSpec       IntervalSpec,
               signature          Signature,
               identitySignature  IdentitySignature
               }

IntervalSpec ::= SEQUENCE {
               chainSpec           ChainSpec,
               intervalStart       GeneralizedTime,
               intervalStop        GeneralizedTime,
               publicKey           PublicKey,
               previousPublicKey   [0] PublicKey  OPTIONAL,
               previousMetaDigest  [1] OCTET STRING  OPTIONAL
```



```
                }

ChainSpec ::= SEQUENCE {
                serverId                  Uri,
                chainStart                GeneralizedTime,
                digestAlgorithm           DigestAlgorithmIdentifier,
                signatureAlgorithm        SignatureAlgorithmIdentifier,
                publicVerificationServer  Uri  OPTIONAL
                }

Uri ::= IA5String

PublicKey ::= OCTET STRING

Signature ::= BIT STRING

IdentitySignature ::= SEQUENCE {
                signatureAlgorithm  SignatureAlgorithmIdentifier,
                signature           Signature,
                certificate         EncodedCertificate  OPTIONAL
                }

EncodedCertificate ::= TYPE-IDENTIFIER.&Type( Certificate )

ArchiveTree ::= SEQUENCE {
                archive   Uri,
                children  ArchiveTreeList  OPTIONAL
                }

ArchiveTreeList ::= SEQUENCE SIZE(1..MAX) OF ArchiveTree

UriList ::= SEQUENCE SIZE(1..MAX) OF Uri

END  -- TransientKey
```

A.20 Module TrustedTimestamp

```
TrustedTimeStamp {
                   iso(1) identified-organization(3) tc68(133) country(16) x9(840)
                      x9Standards(9) x9-95(95) module(0) tts(1) v2010-2016(1) }
DEFINITIONS IMPLICIT TAGS ::= BEGIN
-- EXPORTS All; --
IMPORTS
   -- ISO/IEC 9594-8 | ITU-T Rec. X.509 AuthenticationFramework --
EXTENSION
                FROM AuthenticationFramework {
                joint-iso-itu-t ds(5) module(1) authenticationFramework(7) 8 }
-- ISO/IEC 9594-8 | ITU-T Rec. X.509 CertificateExtensions --
GeneralName
                FROM CertificateExtensions {
                joint-iso-itu-t ds(5) module(1) certificateExtensions(26) 8 }
-- X9.73 Cryptographic Message Syntax (CMS) - ASN.1 and XML --
AuthenticatedData, DigestAlgorithmIdentifier, SignedData
                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) }
-- X9.95 TransientKey -
TransientKeySignedTST, transientKeySignedTST, tsp-req-tk
                FROM TransientKey {
                iso(1) identified-organization(3) tc68(133) country(16) x9(840)
                x9Standards(9) x9-95(95) module(0) tk(2) version(0) v2010-2016(1) };
-- Time stamp Request --
TimeStampReq ::= SEQUENCE {
                version         Version,
                messageImprint  MessageImprint,
                reqPolicy       TSAPolicyId  OPTIONAL,
                nonce           Nonce  OPTIONAL,
                certReq         BOOLEAN  DEFAULT FALSE,
                extensions      [0] Extensions  OPTIONAL
                }
MessageImprint ::= SEQUENCE {
                hashAlgorithm  DigestAlgorithmIdentifier,
```

```
                hashedMessage  OCTET STRING
                }
MessageImprints ::= SEQUENCE SIZE(1..MAX) OF MessageImprint
                TSAPolicyId ::= POLICY.&id({TSAPolicies})
                TSAPolicies POLICY ::= {
                --
                ... -- Any supported TSA policy --
                }
TSAPolicyId ::= POLICY.&id({TSAPolicies})
TSAPolicies POLICY ::= {
                --
                ... -- Any supported TSA policy --
                }
POLICY ::= OIDS -- Supported TSA policies --
                Nonce ::= INTEGER
                -- Time Stamp Response -
                TimeStampResp ::= SEQUENCE {
                status          PKIStatusInfo,
                timeStampToken  TimeStampToken  OPTIONAL
                }
Nonce ::= INTEGER

-- Time Stamp Response -

TimeStampResp ::= SEQUENCE {
                status          PKIStatusInfo,
                timeStampToken  TimeStampToken  OPTIONAL
                }
PKIStatusInfo ::= SEQUENCE {
                status        PKIStatus,
                statusString  PKIFreeText  OPTIONAL,
                failInfo      PKIFailureInfo  OPTIONAL
                }
PKIStatus ::= INTEGER {
                granted                (0), -- request is completely granted
                grantedWithMods        (1), -- modifications were needed, requester is
                                            -- responsible for asserting the differences
                rejection              (2), -- request not fulfilled, the failure code
                                            -- provides additional information
                waiting                (3), -- request not yet processed, requester
                                            -- receives a receipt that the
                                            -- request has been received
                revocationWarning      (4), -- a revocation is imminent
                revocationNotification (5)  -- a revocation has occurred
                }
PKIFreeText ::= SEQUENCE SIZE(1..MAX) OF UTF8String
PKIFailureInfo ::= BIT STRING {
                badAlg                 (0), -- unrecognized or unsupported algorithm
                badRequest             (2), -- transaction not permitted or supported
                badDataFormat          (5), -- data submitted has the wrong format
                timeNotAvailable      (14), -- TSAs service is not available
                unacceptedPolicy      (15), -- requested TSA policy is not supported
                unacceptedExtension   (16), -- requested TSA extension is not supported
                addInfoNotAvailable   (17), -- requested additional info not available
                systemNotAvailable    (24), -- system is not available
                systemFailure         (25), -- system failure
                verificationFailure   (27)  -- verification of time stamp has failed
                }
-- Time stamp token --
TimeStampToken ::= SEQUENCE {
                contentType  CONTENTS.&id({Contents}),
                content      [0] EXPLICIT CONTENTS.&Type({Contents}{@contentType})
                }
Contents CONTENTS ::= {
                { SignedData            IDENTIFIED BY id-signedData } |
                { AuthenticatedData     IDENTIFIED BY id-ct-authData } |
                { DigestedData          IDENTIFIED BY id-digestedData } |
                { TransientKeySignedTST IDENTIFIED BY transientKeySignedTST },
                --
                ...  -- Expect additional time-stamp encapsulations --
```



```
                }

TSTInfo ::= SEQUENCE {
                version         Version,
                policy          TSAPolicyId,
                messageImprint  MessageImprint,
                serialNumber    SerialNumber,
                genTime         GeneralizedTime,
                accuracy        Accuracy  OPTIONAL,
                ordering        BOOLEAN DEFAULT FALSE,
                nonce           Nonce  OPTIONAL,
                tsa             [0] EXPLICIT GeneralName  OPTIONAL,
                extensions      [1] Extensions  OPTIONAL
                }
Version ::= INTEGER { v1(1) }
SerialNumber ::= INTEGER  -- Expect large values --
Accuracy ::= SEQUENCE {
                seconds  INTEGER OPTIONAL,
                millis   [0] INTEGER(1..999)  OPTIONAL,
                micros   [1] INTEGER(1..999)  OPTIONAL
                } (ALL EXCEPT({ -- No components present -- }))
-- TSTInfo encapsulation --
ETSTInfo ::= OCTET STRING (CONTAINING TSTInfo)
id-ct-TSTInfo OID ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                pkcs-9(9) smime(16) ct(1) tstInfo(4) }
EContents CONTENTS ::= {
                { ETSTInfo IDENTIFIED BY id-ct-TSTInfo },
                --
                ... -- Expect additional content types --
                }
EncapsulatedContentInfo::= SEQUENCE {
                eContentType  CONTENTS.&id({EContents}),
                eContent      [0] EXPLICIT CONTENTS.&Type({EContents}{@eContentType})
                }
-- Verification of a time stamp token --
VerifyReq ::= SEQUENCE {
                version    Version,
                tst        TimeStampToken,
                requestID  RequestID  OPTIONAL
                }
VerifyResp ::= SEQUENCE {
                version    Version,
                status     PKIStatusInfo,
                tst        TimeStampToken,
                requestID  RequestID  OPTIONAL
                }
-- Extend operation on a time stamp token -
ExtendReq ::= SEQUENCE {
                version    Version,
                tst        TimeStampToken,
                requestID  [0] OCTET STRING  OPTIONAL
                }
ExtendResp ::= SEQUENCE {
                   version    Version,
                   status     PKIStatusInfo,
                   tst        TimeStampToken,
                   requestID  [0] OCTET STRING  OPTIONAL
                }
RequestID ::= OCTET STRING (SIZE(1..MAX))
-- Time stamping extensions --
Extension{EXTENSION:ExtensionSet} ::= SEQUENCE {
                extnId     EXTENSION.&id({ExtensionSet}),
                critical   BOOLEAN DEFAULT FALSE,
                extnValue  OCTET STRING
                }
Extensions ::= SEQUENCE OF Extension{{TSExtensions}}
TSExtensions EXTENSION ::= {
                extHash    |
                extMethod  |
                extRenewal,
```

```
                --
                ... -- Expect additional extensions --
                }

extHash EXTENSION ::= {
                SYNTAX ExtHash IDENTIFIED BY tsp-ext-hash
                }
ExtHash ::= SEQUENCE SIZE(1..MAX) OF MessageImprint
extMethod EXTENSION ::= {
                SYNTAX ExtMethod IDENTIFIED BY tsp-ext-meth
                }
ExtMethod ::= SEQUENCE SIZE(1..MAX) OF Method
Method ::= METHOD.&id({Methods})
Methods METHOD ::= {
                { OID tsp-itm-ds      } |
                { OID tsp-itm-mac     } |
                { OID tsp-req-link    } |
                { OID tsp-req-link-ds } |
                { OID tsp-req-tk      },
                --
                ... -- Any time stamping method --
                }
extRenewal EXTENSION ::= {
                SYNTAX ExtRenewal IDENTIFIED BY tsp-ext-renewal
                }
ExtRenewal ::= TimeStampToken
tsp-ext-renewal OID ::= {
                iso(1) standard(0) time-stamp(18014) extensions(1) renewal(3) }
-- Information object identifiers
tsp-ext-hash OID ::= {
                iso(1) standard(0) time-stamp(18014) extensions(1) hash(1)}
tsp-ext-meth OID ::= {
                iso(1) standard(0) time-stamp(18014) extensions(1) meth(2) }
tsp-itm-ds OID ::= {
                iso(1) standard(0) time-stamp(18014) itm(2) ds(1)}
tsp-itm-mac OID ::= {
                iso(1) standard(0) time-stamp(18014) itm(2) mac(2)}
tsp-req-link OID ::= {
                iso(1) standard(0) time-stamp(18014) lt(3) link(1)}
tsp-req-link-ds OID ::= {
                iso(1) standard(0) time-stamp(18014) lt(3) link-ds(2)}
id-signedData OID ::= {
                iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 2 }
id-ct-authData OID ::= {
                iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
                pkcs-9(9) smime(16) ct(1) authData(2) }
id-digestedData OID ::= {
                iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs7(7) 5 }
-- Linked token encapsulation object identifiers --
tsp-digestedData OID ::= {
                iso(1) standard(0) time-stamp(18014) lt(3) digestedData(8) }
tsp-signedData OID ::= {
                iso(1) standard(0) time-stamp(18014) lt(3) signedData(9) }
-- Link token types -
DigestedData ::= SEQUENCE {
                version           DDVersion95,
                digestAlgorithm   DigestAlgorithmIdentifier,
                encapContentInfo  EncapsulatedContentInfo,
                digest            Digest
                }
DDVersion95 ::= INTEGER { version2(2) } (version2, ...)
Digest ::= OCTET STRING
BindingInfo ::= SEQUENCE {
                version      Version,
                msgImprints  MessageImprints,
                aggregate    [0] Chains  OPTIONAL,
                links        Links,
                publish      [1] Chains  OPTIONAL,
                extensions   [2] BindingInfoExtensions  OPTIONAL
                }
```



```
Chains ::= SEQUENCE SIZE(1..MAX) OF Chain
Chain ::= SEQUENCE {
               algorithm  ChainAlgorithmIdentifier,
               links      Links
               }
ChainAlgorithmIdentifier ::= AlgorithmIdentifier {{ ChainAlgorithms }}
ChainAlgorithms ALGORITHM ::= {
               --
               ... -- Expect additional chain algorithms --
               }
Links ::= SEQUENCE SIZE(1..MAX) OF Link
Link ::= SEQUENCE {
               algorithm   [0] LinkAlgorithmIdentifier  OPTIONAL,
               identifier  [1] INTEGER  OPTIONAL,
               members      Nodes
               }
LinkAlgorithmIdentifier ::= AlgorithmIdentifier {{ LinkAlgorithms }}
LinkAlgorithms ALGORITHM ::= {
               --
               ... -- Expect additional link algorithms --
               }
Nodes ::= SEQUENCE SIZE(1..MAX) OF Node
Node ::= CHOICE {
               imprints   [0] Imprints,
               reference  [1] INTEGER
               }
Imprints ::=  SEQUENCE SIZE(1..MAX) OF Imprint
Imprint ::= OCTET STRING
-- BindingInfo extensions --
BindingInfoExtensions ::= SEQUENCE OF Extension{{BIExtensions}}
BIExtensions EXTENSION ::= {
               extName        |
               extTime        |
               extPublication,
               --
               ... -- Expect additional extensions --
               }
extName EXTENSION ::= { SYNTAX ExtName IDENTIFIED BY tsp-ext-name }
ExtName ::= GeneralName
tsp-ext-name OID ::= {
               iso(1) standard(0) time-stamp(18014) lt(3) name(5) }
extTime EXTENSION ::= { SYNTAX ExtTime IDENTIFIED BY tsp-ext-time }
ExtTime ::= GeneralizedTime
tsp-ext-time OID ::= {
               iso(1) standard(0) time-stamp(18014) lt(3) time (6) }
extPublication EXTENSION ::= {
               SYNTAX ExtPublication IDENTIFIED BY tsp-ext-publication
               }
ExtPublication ::= SEQUENCE SIZE (1..MAX) OF PublicationInfo
tsp-ext-publication OID ::= {
               iso(1) standard(0) time-stamp(18014) lt(3) publication (7) }
PublicationInfo ::= SEQUENCE {
               pubTime   GeneralizedTime OPTIONAL,
               pubId     [0] GeneralName OPTIONAL,
               pubChains [1] Chains      OPTIONAL,
               sourceId  [2] GeneralName OPTIONAL
               }
-- Merkle chain algorithm object identifier  from Annex E --
id-merkle-chain OID ::= {
               iso(1) identified-organization(3) tc68(133) country(16) x9(840)
               x9Standards(9) x9-95(95) ids(1) merkle-chain(1) }
merkle-chain ALGORITHM ::= {
               OID id-merkle-chain PARMS MerkleChainParms
               }
MerkleChainParms ::= SEQUENCE SIZE(1..MAX) OF HashFunction
HashFunction ::= DigestAlgorithmIdentifier
-- Time calibration --
TimeCalibrationReport ::= SEQUENCE {
               version        Version,
               tseInfo        EntityInfo,
```

```
                tsaInfo         EntityInfo,
                dutInfo         [0] EntityInfo  OPTIONAL,
                timingMetrics   TimingMetrics
                }
EntityInfo ::= SEQUENCE {
                entityName    UTF8String  OPTIONAL,
                entityID      OBJECT IDENTIFIER  OPTIONAL,
                entityOption  OCTET STRING  OPTIONAL
                } (ALL EXCEPT ({ -- None; at least one component shall be present -- }))
TimingMetrics ::= SEQUENCE {
                ntpTime      GeneralizedTime, -- Time at which certification took place
                offset       Accuracy,        -- Current lower clock offset
                delay        Accuracy,        -- Path propagation delay
                leapSecond   LeapSecond  OPTIONAL
                }
LeapSecond ::= SEQUENCE {
                leapDay  GeneralizedTime,
                action   INTEGER(0..1)  -- 1: last minute has 61 seconds --
                                        -- 0: last minute has 59 seconds --
                }
-- Supporting definitions --
OID ::= OBJECT IDENTIFIER -- Alias
OIDS ::= CLASS {
                &id OBJECT IDENTIFIER  UNIQUE
                }
WITH SYNTAX { OID &id }
CONTENTS ::= TYPE-IDENTIFIER -- ISO/IEC 8824-2, Annex A --
AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
                algorithm   ALGORITHM.&id({IOSet}),
                parameters  ALGORITHM.&Type({IOSet}{@algorithm})  OPTIONAL
                }
ALGORITHM ::= CLASS {
                &id    OBJECT IDENTIFIER  UNIQUE,
                &Type  OPTIONAL
                }
WITH SYNTAX { OID &id [PARMS &Type] }
METHOD ::= CLASS {
                &id OBJECT IDENTIFIER  UNIQUE
                }
WITH SYNTAX { OID &id }
END  -- TrustedTimeStamp --
```

A.21 Module ANSI-X9-42

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

DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- X9.42 Diffie-Hellman and MQV Symmetric Key Agreement

-- EXPORTS All;

-- IMPORTS None;

FIELD-ID ::= TYPE-IDENTIFIER

FieldID { FIELD-ID:IOSet } ::= SEQUENCE {
                fieldType FIELD-ID.&id({IOSet}),
                parameters FIELD-ID.&Type({IOSet}{@fieldType})
                }

FiniteFields ::= FieldID {{ FieldTypes }}

FieldTypes FIELD-ID ::= {
                { DomainParameters IDENTIFIED BY gfPrime }, -- GF(p)
                ...
                }

DomainParameters ::= SEQUENCE {  -- Galois field group parameters
                p   INTEGER, -- odd prime, p = jq + 1
                g   INTEGER, -- generator, g ^ q = 1 mod p
                q   INTEGER, -- prime factor of p-1
```



```
                j   INTEGER OPTIONAL,     -- cofactor, j ? 2
                validationParms ValidationParms OPTIONAL
                }

-- Note: The domain parameter cofactor is required when using the cofactor method.

ValidationParms ::= SEQUENCE {
                seed BIT STRING, -- seed for prime number generation
                pGenCounter INTEGER       -- parameter verification
                }

-- Diffie-Hellman Public Number

SubjectPublicKeyInfo ::= SEQUENCE {
                algorithm   AlgorithmIdentifier {{ DHPublicNumbers }},
                subjectPublicKey BIT STRING
                }

DHPublicNumbers ALGORITHM-ID ::= {
                { OID dhPublicNumber PARMS DomainParameters },
                ...
                }

DiffieHellmanPublicNumber ::= INTEGER

-- Key Agreement Schemes

SchemeSyntax { KeyDerivationMethod: kdm } ::= CHOICE {
                schemeId    SchemeIdentifier,
                oid         SchemeOID,
                number      SchemeNumber
                } (CONSTRAINED BY { KeyDerivationMethod: kdm } )

KeyDerivationMethod ::= OBJECT IDENTIFIER

SchemeIdentifier ::= SEQUENCE {
                scheme      KEY-AGREEMENT.&id({Schemes}),
                parameters  KEY-AGREEMENT.&Type({Schemes}{@scheme}) OPTIONAL
                }

Schemes KEY-AGREEMENT ::= {
                { OID dhStatic     PARMS SchemeParameters } |
                { OID dhEphem      PARMS SchemeParameters } |
                { OID dhOneFlow    PARMS SchemeParameters } |
                { OID dhHybrid1    PARMS SchemeParameters } |
                { OID dhHybrid2    PARMS SchemeParameters } |
                { OID dhHybridOneFlow PARMS SchemeParameters } |
                { OID mqv2         PARMS  SchemeParameters } |
                { OID mqv1         PARMS  SchemeParameters } |
                --
                NamedSchemes,
                ...
                }

KEY-AGREEMENT ::= ALGORITHM-ID

SchemeParameters ::= AlgorithmIdentifier {{ KeyDerivationAlgorithm }}

KeyDerivationAlgorithm ALGORITHM-ID ::= {
                { OID  id-sha1  PARMS NULL },   -- From ANS X9.30
                ...
                }

HMAC ::= OCTET STRING

SchemeOID ::= KEY-AGREEMENT.&id({NamedSchemes})

NamedSchemes KEY-AGREEMENT ::= {
                { OID  dhStatic-sha1 } |        -- { dhStatic, {id-sha1,NULL} }
                { OID  dhEphem-sha1  } |        -- { dhEphem, {id-sha1,NULL} }
                { OID dhOneFlow-sha1 } | -- { dhOneFlow, {id-sha1,NULL} }
                { OID  dhHybrid1-sha1 } |-- { dhHybrid1, {id-sha1,NULL} }
                { OID  dhHybrid2-sha1 } |-- { dhHybrid2, {id-sha1,NULL} }
                { OID dhHybridOneFlow-sha1 } | -- { dhHybridOneFlow, {id-sha1,NULL} }
                { OID mqv2-sha1    } |          -- { mqv2,   {id-sha1,NULL} }
```

```
                { OID mqv1-sha1    },           -- { mqv1,   {id-sha1,NULL} }
                ...
                }

SchemeNumber ::= ENUMERATED {
                dhStatic-sha1      (0),         -- { dhStatic, {id-sha1,NULL} }
                dhEphem-sha1       (1),         -- { dhEphem, {id-sha1,NULL} }
                dhOneFlow-sha1     (2),         -- { dhOneFlow, {id-sha1,NULL} }
                dhHybrid1-sha1     (3),         -- { dhHybrid1, {id-sha1,NULL} }
                dhHybrid2-sha1     (4),         -- { dhHybrid2, {id-sha1,NULL} }
                dhHybridOneFlow-sha1 (5),       -- { dhHybridOneFlow, {id-sha1,NULL} }
                mqv2-sha1          (6),         -- { mqv2,   {id-sha1,NULL} }
                mqv1-sha1          (7),         -- { mqv1,   {id-sha1,NULL} }
                ...
                }

-- Per-Party Public Information

OtherInfo ::= SEQUENCE {
                keyInfo     AlgorithmIdentifier {{ KeySpecificAlgorithms }},
                partyUInfo         [0] OCTET STRING OPTIONAL,
                partyVInfo         [1] OCTET STRING OPTIONAL,
                suppPubInfo        [2] OCTET STRING OPTIONAL,
                suppPrivInfo       [3] OCTET STRING OPTIONAL
                }

KeySpecificAlgorithms ALGORITHM-ID ::= {
                { OID tripleDES PARMS Counter },
                ...
                }

Counter ::= OCTET STRING (SIZE (4))

TripleDES ::= SEQUENCE SIZE(oneKey..threeKeys) OF INTEGER

oneKey     INTEGER ::= 1 -- 1-key: for backwards compatability

threeKeys  INTEGER ::= 3 -- 3-key: the best, though 2-key gains

-- most of the Triple DES strength.

-- Object identifiers

ansi-X9-42 OBJECT IDENTIFIER ::= {
               iso(1) member-body(2) us(840) ansi-x942(10046) }

fieldType  OBJECT IDENTIFIER ::= { ansi-X9-42 fieldType(0) }

gfPrime    OBJECT IDENTIFIER ::= { fieldType 1 } -- GF(p)

algorithm  OBJECT IDENTIFIER ::= { ansi-X9-42 algorithms(1) }

tripleDES  OBJECT IDENTIFIER ::= { algorithm triple-des(2) }

numberType OBJECT IDENTIFIER ::= { ansi-X9-42 numberTypes(2) }

dhPublicNumber OBJECT IDENTIFIER ::= {numberType 1 } -- Diffie-Hellman public number

scheme     OBJECT IDENTIFIER ::= { ansi-X9-42 schemes(3) }

dhStatic   OBJECT IDENTIFIER ::= {scheme 1 }   -- Diffie-Hellman, Static Only

dhEphem    OBJECT IDENTIFIER ::= {scheme 2 }   -- Diffie-Hellman, Ephemeral Only

dhOneFlow  OBJECT IDENTIFIER ::= {scheme 3 }   -- Diffie-Hellman, One Flow

dhHybrid1  OBJECT IDENTIFIER ::= {scheme 4 }   -- Diffie-Hellman, Hybrid, One Group

dhHybrid2  OBJECT IDENTIFIER ::= {scheme 5 }   -- Diffie-Hellman, Hybrid, Two Groups

dhHybridOneFlow OBJECT IDENTIFIER ::= {scheme 6 } -- Diffie-Hellman, Hybrid, OneFlow

mqv2       OBJECT IDENTIFIER ::= {scheme 7 }
               -- Menezes-Qu-Vanstone Method, Two pairs/Two pairs

mqv1       OBJECT IDENTIFIER ::= {scheme 8 }
               -- Menezes-Qu-Vanstone Method, Two pairs/one pair

namedScheme    OBJECT IDENTIFIER ::= { ansi-X9-42 names(4) }
```



```
dhStatic-sha1  OBJECT IDENTIFIER ::= { namedScheme 1 }

dhEphem-sha1   OBJECT IDENTIFIER ::= { namedScheme 2 }

dhOneFlow-sha1 OBJECT IDENTIFIER ::= { namedScheme 3 }

dhHybrid1-sha1 OBJECT IDENTIFIER ::= { namedScheme 4 }

dhHybrid2-sha1 OBJECT IDENTIFIER ::= { namedScheme 5 }

dhHybridOneFlow-sha1 OBJECT IDENTIFIER ::= { namedScheme 6 }

mqv2-sha1      OBJECT IDENTIFIER ::= { namedScheme 7 }

mqv1-sha1      OBJECT IDENTIFIER ::= { namedScheme 8 }

keyDerivationMethod        OBJECT  IDENTIFIER  ::=    { ansi-X9-42 kdMethods(5) }

kdasn1der      OBJECT      IDENTIFIER    ::=
               { keyDerivationMethod asn1der(0) }

kdConcatenation   OBJECT IDENTIFIER ::= {keyDerivationMethod concatenation(1)}

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

-- Supporting definitions

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

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

END -- ANSI-X9-42 --
```

A.22 Module ANSI-X9-62

```
ANSI-X9-62 { iso(1) member-body(2) us(840) 10045 module(0) 2 }
DEFINITIONS EXPLICIT TAGS ::= BEGIN
-- EXPORTS All;
-- IMPORTS None;
-- ============================================
-- Notes
-- ============================================
-- 1. Definitions in this module are arranged to minimize forward references,
-- Reading backwards gives a top-down approach more like X9.62-1998.
-- 2. Most comments briefly explain subsequent definition.
-- ============================================
-- Common Object Identifier (see  E.2)
-- ============================================
-- The root OID for this module.
ansi-X9-62 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) 10045 }
-- ============================================
-- Definition for Algorithm Identifiers (see  E.3)
-- ============================================
-- Information object class used to for algorithm identifiers.
-- Note: Original X9.62-1998 was TYPE-IDENTIFIER
-- New version here agrees with X9.63-2001
ALGORITHM ::= CLASS {
               &id  OBJECT IDENTIFIER UNIQUE,
               &Type OPTIONAL
               }
WITH SYNTAX { OID &id [PARMS &Type] }
-- X9.62 profile of a common ASN.1 type AlgorithmIdentifier,
-- The X9.62 version is a parameterized type, to allow restrictions.
AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
               algorithm          ALGORITHM.&id({IOSet}),
               parameters         ALGORITHM.&Type({IOSet}{@algorithm}) OPTIONAL
```

```
                }
-- ==========================================
-- Hash Functions (see  E.4)
-- ==========================================
-- Inherited OID for SHA1
sha-1 OBJECT IDENTIFIER ::= { iso(1)
                identified-organization(3) oiw(14) secsig(3) algorithm(2) 26 }
-- New OID for SHA224
id-SHA224 OBJECT IDENTIFIER ::= {
                joint-iso-itu-t(2)  country(16)  us(840)  organization(1)  gov(101)  csor(3)
                nistalgorithm(4) hashalgs(2) 4 }
-- Inherited OID for SHA256
id-SHA256 OBJECT IDENTIFIER ::= {
                joint-iso-itu-t(2)  country(16)  us(840)  organization(1)  gov(101)  csor(3)
                nistalgorithm(4) hashalgs(2) 1 }
-- Inherited OID for SHA384
id-SHA384 OBJECT IDENTIFIER ::= {
                joint-iso-itu-t(2)  country(16)  us(840)  organization(1)  gov(101)  csor(3)
                nistalgorithm(4) hashalgs(2) 2 }
-- Inherited OID for SHA512
id-SHA512 OBJECT IDENTIFIER ::= {
                joint-iso-itu-t(2)  country(16)  us(840)  organization(1)  gov(101)  csor(3)
                nistalgorithm(4) hashalgs(2) 3 }
-- Information object set of Approved hash functions
ANSIX9HashFunctions ALGORITHM ::= {
                { OID sha-1 } |
                { OID sha-1 PARMS NULL } |
                { OID id-SHA224 } |
                { OID id-SHA224 PARMS NULL } |
                { OID id-SHA256 } |
                { OID id-SHA256 PARMS NULL } |
                { OID id-SHA384 } |
                { OID id-SHA384 PARMS NULL } |
                { OID id-SHA512 } |
                { OID id-SHA512 PARMS NULL } ,
                ... -- Additional hash functions may be added
                }
-- Type (parameterized) to indicate the hash function with
-- the OID ecdsa-with-Specified
HashAlgorithm::= AlgorithmIdentifier {{ ANSIX9HashFunctions }}
-- ==========================================
-- Finite Field Identification (see  E.5)
-- ==========================================
-- Finite field element
FieldElement ::= OCTET STRING
-- Root OID for identifying field types
id-fieldType OBJECT IDENTIFIER ::= { ansi-X9-62 fieldType(1) }
-- OID identifying prime field types
prime-field OBJECT IDENTIFIER ::= { id-fieldType prime(1) }
-- Parameters for prime field
Prime-p ::= INTEGER -- Finite field F(p), where p is an odd prime
-- OID for identifying binary field
characteristic-two-field OBJECT IDENTIFIER ::= {
                id-fieldType characteristic-two(2) }
-- Root OID for identifying binary field basis types
id-characteristic-two-basis OBJECT IDENTIFIER ::= {
                characteristic-two-field basisType(3) }
-- OID to identify a Gaussian normal basis.
gnBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis gaussian(1) }
-- OID to identify a trinomial basis.
tpBasis  OBJECT IDENTIFIER ::= { id-characteristic-two-basis trinomial(2) }
-- Trinomial basis representation of F2^m
-- Integer k for reduction polynomial x^m + x^k + 1
Trinomial ::= INTEGER
-- OID to identify a pentanomial basis.
ppBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis pentanomial(3) }
-- Pentanomial basis representation of F2^m
-- reduction polynomial integers k1, k2, k3
-- f(x) = x^m + x^k3 + x^k2 + x^k1 + 1
Pentanomial ::= SEQUENCE {
```



```
                k1  INTEGER,
                k2  INTEGER,
                k3  INTEGER
                }
-- The object class for binary field basis types
CHARACTERISTIC-TWO ::= TYPE-IDENTIFIER
-- Allowable basis types are given the following info object set
BasisTypes CHARACTERISTIC-TWO ::= {
                { NULL              IDENTIFIED BY  gnBasis } |
                { Trinomial         IDENTIFIED BY  tpBasis } |
                { Pentanomial       IDENTIFIED BY  ppBasis },
                ... -- Additional basis types may be added
                }
-- Parameters for a binary field
Characteristic-two ::= SEQUENCE {
                m                   INTEGER, -- Field size is 2^m
                basis       CHARACTERISTIC-TWO.&id({BasisTypes}),
                parameters          CHARACTERISTIC-TWO.&Type({BasisTypes}{@basis})
                }
-- Information object class used to constrain fields
FIELD-ID ::= TYPE-IDENTIFIER -- ISO/IEC 8824-2:1995(E), Annex A
-- Field types are constrained with this information object set
FieldTypes FIELD-ID ::= {
                { Prime-p  IDENTIFIED BY  prime-field } |
                { Characteristic-two  IDENTIFIED BY  characteristic-two-field },
                ... -- Additional field types may be added
                }
-- Finite fields have a type (prime or binary) and parameters (size and basis)
FieldID { FIELD-ID:IOSet } ::= SEQUENCE {-- Finite field
                fieldType           FIELD-ID.&id({IOSet}),
                parameters          FIELD-ID.&Type({IOSet}{@fieldType})
                }
-- ===========================================
-- Elliptic Curve Points (see  E.6)
-- ===========================================
ECPoint ::= OCTET STRING
-- ===========================================
-- Elliptic Curve Domain Parameters (see  E.7)
-- ===========================================
-- Identifying an elliptic curve by its coefficients (and optional seed)
Curve ::= SEQUENCE {
                a     FieldElement, -- Elliptic curve coefficient a
                b     FieldElement, -- Elliptic curve coefficient b
                seed BIT STRING OPTIONAL
                -- Shall be present if used in SpecifiedECDomain with version of
                -- ecdpVer2 or ecdpVer3
                }
-- Type used to control version of EC domain parameters
SpecifiedECDomainVersion ::= INTEGER { ecdpVer1(1) , ecdpVer2(2) , ecdpVer3(3) }
-- Identifying elliptic curve domain parameters explicitly with this type
SpecifiedECDomain ::= SEQUENCE {
                version             SpecifiedECDomainVersion ( ecdpVer1 | ecdpVer2 | ecdpVer3
                ),
                fieldID             FieldID {{FieldTypes}},
                curve       Curve,
                base                ECPoint, -- Base point G
                order       INTEGER, -- Order n of the base point
                cofactor            INTEGER OPTIONAL, -- The integer h = #E(Fq)/n
                hash                HashAlgorithm OPTIONAL,
                ... -- Additional parameters may be added
                }
-- Arc in X9.62 for naming EC domain parameters that are not named elsewhere
ellipticCurve OBJECT IDENTIFIER ::= { ansi-X9-62 curves(3) }
-- Arc in X9.62 for identifying prime order elliptic curve domain parameters
primeCurve OBJECT IDENTIFIER ::= { ellipticCurve prime(1) }
-- Arc from SEC2 that names EC domain parameters and is used again in X9.62
secgCurve OBJECT IDENTIFIER ::= { iso(1) identified-organization(3)
                certicom(132) curve(0) }
-- Named EC domain parameters in X9.62
ansix9t163k1 OBJECT IDENTIFIER ::= {secgCurve  1 }
```

```
ansix9t163r1 OBJECT IDENTIFIER ::= {secgCurve  2 }
ansix9t163r2 OBJECT IDENTIFIER ::= {secgCurve  15 }
ansix9t193r1 OBJECT IDENTIFIER ::= {secgCurve  24 }
ansix9t193r2 OBJECT IDENTIFIER ::= {secgCurve  25 }
ansix9t233k1 OBJECT IDENTIFIER ::= {secgCurve  26 }
ansix9t233r1 OBJECT IDENTIFIER ::= {secgCurve  27 }
ansix9t239k1 OBJECT IDENTIFIER ::= {secgCurve  3 }
ansix9t283k1 OBJECT IDENTIFIER ::= {secgCurve  16 }
ansix9t283r1 OBJECT IDENTIFIER ::= {secgCurve  17 }
ansix9t409k1 OBJECT IDENTIFIER ::= {secgCurve  36 }
ansix9t409r1 OBJECT IDENTIFIER ::= {secgCurve  37 }
ansix9t571k1 OBJECT IDENTIFIER ::= {secgCurve  38 }
ansix9t571r1 OBJECT IDENTIFIER ::= {secgCurve  39 }
ansix9p160k1 OBJECT IDENTIFIER ::= {secgCurve  9 }
ansix9p160r1 OBJECT IDENTIFIER ::= {secgCurve  8 }
ansix9p160r2 OBJECT IDENTIFIER ::= {secgCurve  30 }
ansix9p192k1 OBJECT IDENTIFIER ::= {secgCurve  31 }
ansix9p192r1 OBJECT IDENTIFIER ::= {primeCurve  1 }
ansix9p224k1 OBJECT IDENTIFIER ::= {secgCurve  32 }
ansix9p224r1 OBJECT IDENTIFIER ::= {secgCurve  33 }
ansix9p256k1 OBJECT IDENTIFIER ::= {secgCurve  10 }
ansix9p256r1 OBJECT IDENTIFIER ::= {primeCurve  7 }
ansix9p384r1 OBJECT IDENTIFIER ::= {secgCurve  34 }
ansix9p521r1 OBJECT IDENTIFIER ::= {secgCurve  35 }
-- The object class and syntax for naming elliptic curve domain parameters.
ECDOMAIN ::= CLASS {
               &id OBJECT IDENTIFIER UNIQUE
               }
WITH SYNTAX { ID &id }
-- Information object set for named elliptic curve domain parameter
ANSIX9NamedDomains ECDOMAIN ::= {
               { ID ansix9t163k1 } |     --  L.5.2.2
               { ID ansix9t163r2 } |     --  L.5.2.3
               { ID ansix9t233k1 } |     --  L.5.3.2
               { ID ansix9t233r1 } |     --  L.5.3.3
               { ID ansix9t283k1 } |     --  L.5.4.2
               { ID ansix9t283r1 } |     --  L.5.4.3
               { ID ansix9t409k1 } |     --  L.5.5.2
               { ID ansix9t409r1 } |     --  L.5.5.3
               { ID ansix9t571k1 } |     --  L.5.6.2
               { ID ansix9t571r1 } |     --  L.5.6.3
               { ID ansix9p192k1 } |     --  L.6.2.2
               { ID ansix9p192r1 } |     --  L.6.2.3
               { ID ansix9p224k1 } |     --  L.6.3.2
               { ID ansix9p224r1 } |     --  L.6.3.3
               { ID ansix9p256k1 } |     --  L.6.4.2
               { ID ansix9p256r1 } |     --  L.6.4.3
               { ID ansix9p384r1 } |     --  L.6.5.2
               { ID ansix9p521r1 } ,     --  L.6.6.2
               ... -- Additional named EC domain parameters may be added.
               }
-- Type for identifying elliptic curve domain parameters
ECDomainParameters ::= CHOICE {
               specified          SpecifiedECDomain, -- Full specification
               named       ECDOMAIN.&id({ANSIX9NamedDomains}), -- Named
               implicitCA         NULL -- Parameters same as issuer CA
               }
-- ==========================================
-- Elliptic Curve Digital Signatures (see  E.8)
-- ==========================================
-- Format for an actual signature
ECDSA-Sig-Value ::= SEQUENCE {
               r   INTEGER,
               s   INTEGER
               }
-- Root OID to identify types of signatures
id-ecSigType OBJECT IDENTIFIER ::= { ansi-X9-62 signatures(4) }
-- Original X9.62-1998 OID for ECDSA
ecdsa-with-Sha1 OBJECT IDENTIFIER ::= {id-ecSigType sha1(1)}
-- New OID indicating the message digest to be the natural size hash
```



```
-- Note: the natural size hash is strongly recommended
ecdsa-with-Recommended OBJECT IDENTIFIER ::= {id-ecSigType recommended(2)}
-- New OID that indicates the message digest to be specified by the parameters
ecdsa-with-Specified OBJECT IDENTIFIER ::= {id-ecSigType specified(3)}
-- New OIDs that indicates the message digest directly
ecdsa-with-Sha224 OBJECT IDENTIFIER ::= {ecdsa-with-Specified 1}
ecdsa-with-Sha256 OBJECT IDENTIFIER ::= {ecdsa-with-Specified 2}
ecdsa-with-Sha384 OBJECT IDENTIFIER ::= {ecdsa-with-Specified 3}
ecdsa-with-Sha512 OBJECT IDENTIFIER ::= {ecdsa-with-Specified 4}
-- An information object set used to constrain ECC algorithms
ECCAlgorithmSet ALGORITHM ::= {
               {OID ecdsa-with-Sha1} |
               {OID ecdsa-with-Sha1 PARMS NULL} |
               {OID ecdsa-with-Recommended} |
               {OID ecdsa-with-Recommended PARMS NULL} |
               {OID ecdsa-with-Specified PARMS HashAlgorithm }  |
               {OID ecdsa-with-Sha224} |
               {OID ecdsa-with-Sha256} |
               {OID ecdsa-with-Sha384} |
               {OID ecdsa-with-Sha512},
               ... -- More ECC algorithms might be added, including key agreement.
               }
-- A type identifying an ECC algorithm
ECCAlgorithm ::= AlgorithmIdentifier {{ECCAlgorithmSet}}
-- A type identifying one or more ECC algorithms with possible preference
ECCAlgorithms ::= SEQUENCE OF ECCAlgorithm
-- ===========================================
-- Elliptic Curve Public Keys (see  E.9)
-- ===========================================
-- Root OID for identifying types of public keys for X9.62
id-publicKeyType OBJECT IDENTIFIER  ::= { ansi-X9-62 keyType(2) }
-- Original X9.62-1998 OID for identifying unrestricted EC public key
id-ecPublicKey OBJECT IDENTIFIER ::= {
               id-publicKeyType unrestricted(1)
               }
-- Algorithm identifier (original X9.62-1998) for
-- EC public key without restrictions
ecPublicKeyType ALGORITHM ::= {
               OID id-ecPublicKey PARMS DomainParameters
               }
-- New OID for identifying EC public key with algorithm restrictions
id-ecPublicKeyRestricted OBJECT IDENTIFIER ::= {
               id-publicKeyType restricted(2)
               }
-- Type identified by id-ecPublicKeyRestricted
ECPKRestrictions ::= SEQUENCE {
               ecDomain           ECDomainParameters,
               -- Identifies the EC domain parameters
               eccAlgorithms      ECCAlgorithms -- Lists the algorithms supported
               -- for this public key
               }
-- Algorithm identifier (new) with feature to restrict algorithm usage.
ecPublicKeyTypeRestricted ALGORITHM ::= {
               OID id-ecPublicKeyRestricted PARMS ECPKRestrictions
               }
-- Information object set of allowable algorithm identifiers
-- in a SubjectPublicKeyInfo of a certificate
ECPKAlgorithms ALGORITHM ::= {
               ecPublicKeyType |
               ecPublicKeyTypeRestricted,
               ... -- Additional algorithm identifiers may be added
               }
-- An instantiation of Alg. Id.
ECPKAlgorithm ::= AlgorithmIdentifier {{ ECPKAlgorithms}}
-- X9.62 profile of an X.509 (ASN.1) field contained
-- in X.509 and PKIX certificates
SubjectPublicKeyInfo ::= SEQUENCE {
               algorithm          ECPKAlgorithm,
               subjectPublicKey   BIT STRING
               }
```

```
END -- ANSI X9.62
```


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 ASN.1 CMS attribute module:

```
       {itu-t      recommendation(0)      x(24)      cms-profile(894)      module(0)
       cMSProfileAttributes(3) version1(1)}

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

e) 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"
```

f) object idenfifier 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"
```

i) object identifier associated to contentLocations attribute

```
{itu-t recommendation(0) x(24)cms-profile(894) attribute(2)
contentLocations(3)}
"/ITU-T/Recommendation/X/CMS-Profile/Attribute/ContentLocations"
```

j) object identifier associated to precedingBlock attribute

```
{itu-t recommendation(0) x(24)cms-profile(894) attribute(2) precedingBlock(4)}
"/ITU-T/Recommendation/X/CMS-Profile/Attribute/PrecedingBlock"
```

k) object identifier associated to timeStamped attribute

```
{itu-t recommendation(0) x(24)cms-profile(894) attribute(2) timeStamped(5)}
"/ITU-T/Recommendation/X/CMS-Profile/Attribute/TimeStamped"
```

l) object identifier associated to sidechains attribute

```
{itu-t recommendation(0) x(24)cms-profile(894) attribute(2) sidechains(6)}
"/ITU-T/Recommendation/X/CMS-Profile/Attribute/Sidechains"
```

m) object identifier associated to parentBlock attribute

```
{itu-t recommendation(0) x(24)cms-profile(894) attribute(2) parentBlock(7)}
"/ITU-T/Recommendation/X/CMS-Profile/Attribute/ParentBlock"
```

Bibliography

[1] Recommendation ITU-T X.680 (2015) | ISO/IEC 8824-1:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*.

[2] Recommendation ITU-T X.681 (2015) | ISO/IEC 8824-2:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*.

[3] Recommendation ITU-T X.682 (2015) | ISO/IEC 8824-3:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Constraint specification*.

[4] Recommendation ITU-T X.683 (2015) | ISO/IEC 8824-4:2015, *Information technology – Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*.

[5] 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)*.

[6] Recommendation ITU-T X.691 (2015) | ISO/IEC 8825-2:2015, *Information technology – ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*.

[7] Recommendation ITU-T X.693 (2015) | ISO/IEC 8825-4:2015, *Information technology – ASN.1 encoding rules: XML Encoding Rules (XER)*.

[8] Recommendation ITU-T X.696 (2015) | ISO/IEC 8825-7:2014, *Information technology – ASN.1 encoding rules: Specification of Octet Encoding Rules (OER)*.

[9] ANSI X9.42-2003, *Public key cryptography for the financial services industry: Agreement of symmetric keys using discrete logarithm cryptography*.

[10] ANSI X9.62-2005, *Public key cryptography for the financial services industry: The elliptic curve digital signature algorithm (ECDSA)*.

[11] ANSI X9.69-2017, *Framework for key management extensions*.

[12] ANSI X9.73-2017, *Cryptographic message syntax – ASN.1 and XML*.

[13] ANSI X9.95-2016, *Trusted time stamp management and security*.

[14] IETF RFC 5911 (2010), *New ASN.1 modules for cryptographic message syntax (CMS) and S/MIME*.

[15] IETF RFC 5912 (2010), *New ASN.1 modules for the public key infrastructure using X.509 (PKIX)*.

[16] IETF RFC 6268 (2011), *Additional New ASN.1 Modules for the Cryptographic Message Syntax (CMS) and the Public Key Infrastructure Using X.509 (PKIX)*.

ICS 35.100.60

Price based on 98 pages