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.