CMS Long-Term Signature Profile Version 1.0

March 2006

Next Generation Electronic Commerce Promotion Council of Japan (ECOM)

Introduction

The following documents define specifications for long-term signature formats:

- ETSI TS 101 733 V1.6.3 (2005-09), "Electronic Signature Formats"
- IETF RFC3126, "Electronic Signature Formats for long term electronic signatures"

The above standards contain many optional definitions, and it is sufficient if only a particular subset of these are implemented in order to ensure the long-term validity of electronically signed documents. In the report entitled "Guidelines concerning Long-Term Storage of Electronically Signed Documents (March 2002)," ECOM outlined the draft of a recommended profile that defines this particular subset required for ensuring the long-term validity of electronically signed documents.

This profile draft further develops and refines the previous profile draft, endeavoring to define only non-redundant items that are necessary for implementing systems that ensure the long-term validity of electronically signed documents. Section 1 provides details of the profile draft, and Section 2 contains a table that summarizes the profile.

1. Details of the CMS long-term signature profile

This profile draft is based on the latest "CMS Advanced Electronic Signatures (CAdES)" specification, and aims to be efficient (limited, as far as possible, to only the required data and time-stamps) and effective (items required to ensure long-term authenticity).

1.1. Signature format

2 basic forms of electronic signatures may be used, BES (Basic Electronic Signature) as shown in Figure 1, and EPES (Explicit Policy Electronic Signature) as shown in Figure 2.

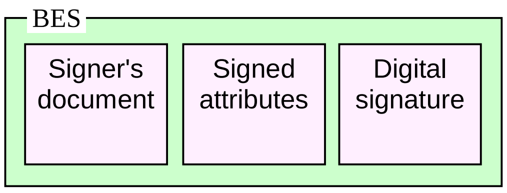

Figure 1: BES

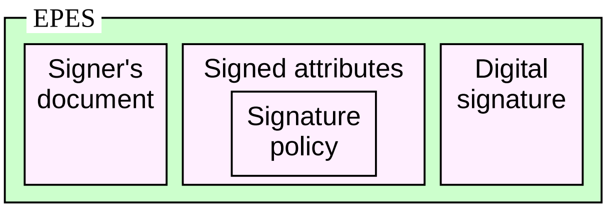

Figure 2: EPES

The difference between these lies in the existence of a signature policy. A signature policy specifies a set of rules relating to the generation and verification of signatures so that a signer and a validator can deem a digital signature to be valid. These rules are set out in "ETSI TR 102 272 V1.1.1 (2003.12): Electronic Signatures and Infrastructures (ESI); ASN.1 format for signature policies," and "RFC3125: Electronic Signature Policies."

Properly formed electronically signed documents conform to the following specifications:

- Cryptographic Message Syntax (CMS: RFC3852)
- Enhanced Security Services (ESS: RFC2634)

(1) General syntax

The general syntax of electronically signed documents is defined in CMS (RFC3852).

```
ContentInfo:
   ContentInfo ::= SEQUENCE {
            contentType  ContentType,
            content     [0] EXPLICIT ANY DEFINED BY contentType }

   ContentType ::= OBJECT IDENTIFIER
```

(2) Data content type

Data content type is as defined in CMS (RFC3852).

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

(3) Signed-data content type

Signed-data content type is as defined in CMS (RFC3852).

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

(4) SignedData type

The syntax of SignedData is as defined in CMS (RFC3852).

```
   SignedData ::= SEQUENCE {
           Version        CMSVersion,
           digestAlgorithms DigestAlgorithmIdentifiers,
           encapContentInfo EncapsulatedContentInfo,
           certificates    [0] IMPLICIT CertificateSet OPTIONAL,
           crls           [1] IMPLICIT CertificateRevocationLists OPTIONAL,
           signerInfos     SignerInfos }

   DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier

   SignerInfos ::= SET OF SignerInfo
```

- SignedData **does not have to be version 3**.
    - ➢ If any of the following are satisfied, the version number is 3:
        - ✧ there is a certificates attribute, and a version 1 attribute certificate, but a version 2 certificate attribute is not present
        - ✧ encapsulated content type is other than id-data
        - ✧ any of the SignerInfo are version 3
    - ➢ However, this excludes cases where there is a certificates attribute and a certificate of a different type is present, or cases where there is a crls attribute and a CRL of a different type is present.

(5) EncapsulatedContentInfo type

The EncapsulatedContentInfo type is as defined in CMS (RFC3852).

```
    EncapsulatedContentInfo ::= SEQUENCE {
                eContentType ContentType,
                eContent    [0] EXPLICIT OCTET STRING OPTIONAL }

    ContentType ::= OBJECT IDENTIFIER
```

- For long-term storage, it is recommended that eContent is contained in SignedData, or stored and managed separately.

(6) SignerInfo type

The SignerInfo type is as defined in CMS (RFC3852).

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

    SignedAttributes ::= SET SIZE (1..MAX) OF Attribute

    UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute

    Attribute ::= SEQUENCE {
                attrType       OBJECT IDENTIFIER,
                attrValues     SET OF AttributeValue }

    AttributeValue ::= ANY

    SignatureValue ::= OCTET STRING
```

- Where there is only one signer, one SignerInfo is sufficient, and where there are several signatures attached in parallel, several SignerInfo values may be created.
- The SignerInfo **version is not an issue of concern**:
  - ➢ If SignerIdentifier is issuerAndSerialNumber, the version is 1
  - ➢ If SignerIdentifier is subjectKeyIdentifier the version is 3
- For long-term signatures, SignedAttributes must contain at least the following values:
  - ➢ ContentType
  - ➢ MessageDigest
  - ➢ SigningCertificate

(i) Message digest calculation process

As defined in CMS (RFC3852).

(ii) Message signature generation process

As defined in CMS (RFC3852).

(iii) Message signature verification process

The procedures for message signature verification are defined in CMS (RFC3852) and enhanced in the present document.

The signature verification process shall use the signer's public key, which shall be verified as correct using the ESS signing-certificate attribute, or the Other-signing-certificate attribute.

(7) Mandatory CMS attributes

The following attributes must be present in the signed attributes of the signature data.

(i) Content type

Syntax is as defined in CMS (RFC3852).

```
   id-contentType OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                          rsadsi(113549) pkcs(1)
                                          pkcs9(9) 3 }

   ContentType ::= OBJECT IDENTIFIER
```

- The signed attributes must contain one and only one ContentType attribute.

(ii) Message digest

Syntax is as defined in CMS (RFC3852).

```
   Id-messageDigest OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                          rsadsi(113549) pkcs(1)
                                          pkcs9(9) 4 }

   MessageDigest ::= OCTET STRING
```

- The value of the MessageDigest attribute is that required by the V part of the ASN.1 TLV of the eContent OCTET STRING within encapContentInfo using the DigestAlgorithm in SignerInfo.
  The signed attributes must contain one and only one MessageDigest attribute.

(iii) The Signing certificate attribute

signing-data must contain only one signing certificate attribute, either ESS signing-certificate or Other-signing-certificate. This attribute value is used to prevent simple substitution attacks and re-issue attacks.

A) ESS signing certificate attribute definition

The ESS signing-certificate attribute is based on ESS (RFC2634). ESS signing-certificate must be a signed attribute.

The signing certificate attribute, or the Other signing certificate attribute described in the following clause must be present. The attribute must be present, and the value must not be empty. The certificate used for signature verification must be obtained from the value of this attribute. The signature validation policy may mandate other certificates be present that may include all the certificates up to the point of trust. ESSCertID must contain an issuerSerial field.

The issuerAndSerialNumber present in the SignerInfo shall be consistent with the issuerSerial field. The certificate identified by ESSCertID shall be used during the signature verification process. If the hash of the certificate does not match the certificate used to verify the signature, the signature shall be considered invalid.

The policy information field is not used.

```
SigningCertificate ::=  SEQUENCE {
             certs       SEQUENCE OF ESSCertID,
             policies     SEQUENCE OF PolicyInformation OPTIONAL -- not
             used
}

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

ESSCertID ::=  SEQUENCE {
             certHash       Hash,
             issuerSerial    IssuerSerial OPTIONAL
}

Hash ::= OCTET STRING -- SHA1 hash of entire certificate

IssuerSerial ::= SEQUENCE {
             Issuer         GeneralNames,
             serialNumber    CertificateSerialNumber
   }
```

B) Other signing certificate attribute definition

This is identical to the ESS SigningCertificate defined above except that this attribute can be used with hashing algorithms other than SHA-1.

```
id-aa-ets-otherSigCert OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840)
                      rsadsi(113549) pkcs(1) pkcs9(9) smime(16)
                      id-aa(2) 19 }

OtherSigningCertificate ::=  SEQUENCE {
          certs       SEQUENCE OF OtherCertID,
          policies     SEQUENCE OF PolicyInformation OPTIONAL -- not
          used
   }

OtherCertID ::= SEQUENCE {
          otherCertHash          OtherHash,
          issuerSerial           IssuerSerial OPTIONAL
   }

OtherHash ::= CHOICE {
          sha1Hash  OtherHashValue,  -- this contains a hash if SHA-1
          is used
          otherHash  OtherHashAlgAndValue
   }

OtherHashValue ::= OCTET STRING

OtherHashAlgAndValue ::= SEQUENCE {
          hashAlgorithm  AlgorithmIdentifier,
          hashValue     OtherHashValue
   }
```

* It must be possible to process both the ESS SigningCertificate attribute and the Other signing certificate attributes at the time of verification.

(8) Mandatory attributes for EPES

(i) Signature policy identifier

For EPES a reference to the signature policy must be included. The signature policy identifier must be a signed attribute.

```
id-aa-ets-sigPolicyId OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                         rsadsi(113549) pkcs(1) pkcs9(9) smime(16)
                         id-aa(2) 15 }

SignaturePolicyIdentifier ::= CHOICE{
         SignaturePolicyId        SignaturePolicyId,
         SignaturePolicyImplied    SignaturePolicyImplied }

SignaturePolicyId ::= SEQUENCE {
         sigPolicyIdentifier   SigPolicyId,
         sigPolicyHash        SigPolicyHash,
         sigPolicyQualifiers   SEQUENCE SIZE (1..MAX) OF
            SigPolicyQualifierInfo     OPTIONAL
   }

SignaturePolicyImplied ::= NULL

SigPolicyId ::= OBJECT IDENTIFIER

SigPolicyHash ::= OtherHashAlgAndValue

SigPolicyQualifierInfo ::= SEQUENCE {
         sigPolicyQualifierId  SigPolicyQualifierId,
         sigQualifier        ANY DEFINED BY sigPolicyQualifierId
   }

SigPolicyQualifierId ::=  OBJECT IDENTIFIER
id-spq-ets-uri OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                         rsadsi(113549) pkcs(1) pkcs9(9) smime(16)
                         id-spq(5) 1 }

SPuri ::= IA5String

id-spq-ets-unotice OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                         rsadsi(113549) pkcs(1) pkcs9(9) smime(16)
                         id-spq(5) 2 }

SPUserNotice ::= SEQUENCE {
         noticeRef       NoticeReference OPTIONAL,
         explicitText     DisplayText OPTIONAL
   }

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

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

(9) Optional CMS attributes

The following attributes may be present in the signed data as defined by this document, and the presence of these attributes must not produce an error at build time or verification time.

(i) Signing time

Syntax is as defined in CMS (RFC3852), however, use of GeneralizedTime (expressed in YYYYMMDDHHMMSSZ format; fractions of a second are not used), rather than UTCTime, is recommended.

```
id-signingTime OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                        rsadsi(113549) pkcs(1)
                                        pkcs9(9) 5 }

SigningTime ::= Time

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

- The signed attributes must not contain more than one SigningTime attribute.

(ii) Countersignature

Countersignature must be an unsigned attribute.

```
id-countersignature OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                 rsadsi(113549) pkcs(1) pkcs9(9) 6 }

Countersignature ::= SignerInfo
```

* There is no limitation placed on the time of generation of countersignature attributes included in the unsigned attributes (essentially, countersignatures may be attached to signatures after the generation of an ES-A). This may also be applied to the long-term signature format. Caution is required regarding the target of archiveTimeStamps of signatures counter-signed by countersignatures.

(10) Optional ESS attributes

The following attributes may be present in the signed data as defined by this document, and the presence of these attributes must not produce an error at build time or verification time.

(i) The content reference attribute

ContentReference is a signed attribute.

```
id-aa-contentReference   OBJECT IDENTIFIER ::= { iso(1) member-body(2)
                  us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                  id-aa(2) 10 }

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

(ii) The ContentIdentifier attribute

ContentIdentifier is a signed attribute.

```
   id-aa-contentReference   OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840)
                            rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                            id-aa(2) 10 }

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

(iii) The ContentHints attribute

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

   id-aa-contentHint OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
            rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 4}
```

(11) Other optional attributes

The following attributes may be present in the signed data as defined by this document, and the presence of these attributes must not produce an error at build time or verification time.

(i) The CommitmentTypeIndication attribute

CommitmentTypeIndication is a signed attribute.

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

   CommitmentTypeIndication ::= SEQUENCE {
            commitmentTypeId          CommitmentTypeIdentifier,
            commitmentTypeQualifier    SEQUENCE SIZE (1..MAX) OF
                                         CommitmentTypeQualifier
                                         OPTIONAL
      }

   CommitmentTypeIdentifier ::= OBJECT IDENTIFIER

   CommitmentTypeQualifier ::= SEQUENCE {
            commitmentTypeIdentifier   CommitmentTypeIdentifier,
            qualifier               ANY DEFINED BY
            commitmentTypeIdentifier
      }
```

id-cti-ets-proofOfOrigin OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 1}
id-cti-ets-proofOfReceipt OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 2}
id-cti-ets-proofOfDelivery OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 3}
id-cti-ets-proofOfSender OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 4}
id-cti-ets-proofOfApproval OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 5}
id-cti-ets-proofOfCreation OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 6}

Proof of origin	Indicates that the signer recognizes to have created, approved and sent the signed data object.
Proof of receipt	Indicates that signer recognizes to have received the content of the signed data object.
Proof of delivery	Indicates that the TSP (Trusted Service Provider) providing that indication has delivered a signed data object in a local store accessible to the recipient of the signed data object.
Proof of sender	Indicates that the entity providing that indication has sent the signed data object (but not necessarily created it).
Proof of approval	Indicates that the signer has approved the content of the signed data object.
Proof of creation	Indicates that the signer has created the signed data object (but not necessarily approved, nor sent it).

(ii) The SignerLocation attribute

SignerLocation is a signed attribute.

```
id-aa-ets-signerLocation OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840)
                         rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                         id-aa(2) 17}

SignerLocation ::= SEQUENCE {
            -- at least one of the following shall be present
         countryName        [0] DirectoryString     OPTIONAL,
            -- As used to name a Country in X.500
         localityName       [1] DirectoryString     OPTIONAL,
            -- As used to name a locality in X.500
         postalAdddress      [2] PostalAddress       OPTIONAL
   }

PostalAddress ::= SEQUENCE SIZE(1..6) OF DirectoryString
```

(iii) The SignerAttributes attribute

SignerAttributes is a signed attribute.

```
id-aa-ets-signerAttr OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                         rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                         id-aa(2) 18}

SignerAttribute ::= SEQUENCE OF CHOICE {
         claimedAttributes     [0]  ClaimedAttributes,
         certifiedAttributes    [1]  CertifiedAttributes
   }

ClaimedAttributes ::= SEQUENCE OF Attribute

CertifiedAttributes ::= AttributeCertificate
```

(iv) The ContentTimeStamp attribute

ContentTimeStamp is a signed attribute.

```
id-aa-ets-contentTimestamp OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840)
                         rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                         id-aa(2) 20}

ContentTimestamp::= TimeStampToken
```

(12) Support for multiple signatures

(i) Independent signatures

Independent signatures are used when multiple persons sign the same document in parallel, and multiple signatures are therefore attached. Each individual signature is independent, and this is achieved by using multiple SignerInfo sequences. Each SignerInfo may independently apply the long-term signature format.

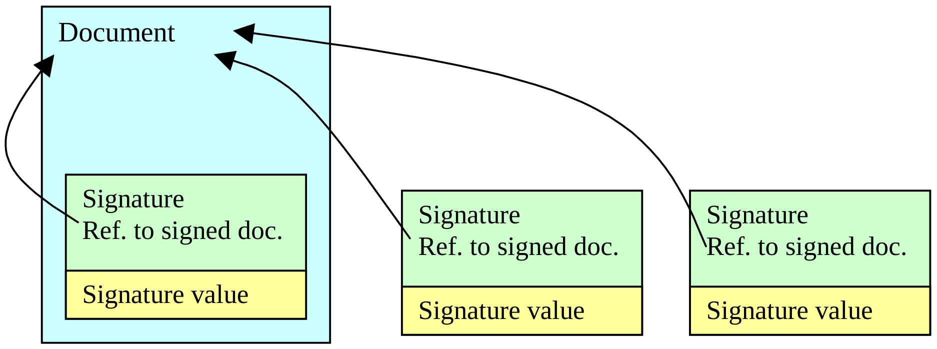

Figure 3: Independent signatures

(ii) Embedded signatures

Embedded signatures are applied one after the other. The counterSignature attribute is used to achieve this. There is no limitation placed on the time of generation of countersignature attributes (countersignatures may also be attached to signatures after the generation of an ES-A). This may also be applied to the long-term signature format. Caution is required regarding the target of archiveTimeStamps of signatures counter-signed by countersignatures.

With regards to the application of countersignatures to long-term signature formats, the signed item is not the main document (the 'Document' in Figure 4), but instead it is only the signature value that contains the hash value of the main document. Careful examination to determine its validity is therefore required. This profile does not place any restrictions on the application of countersignatures to long-term signature formats.

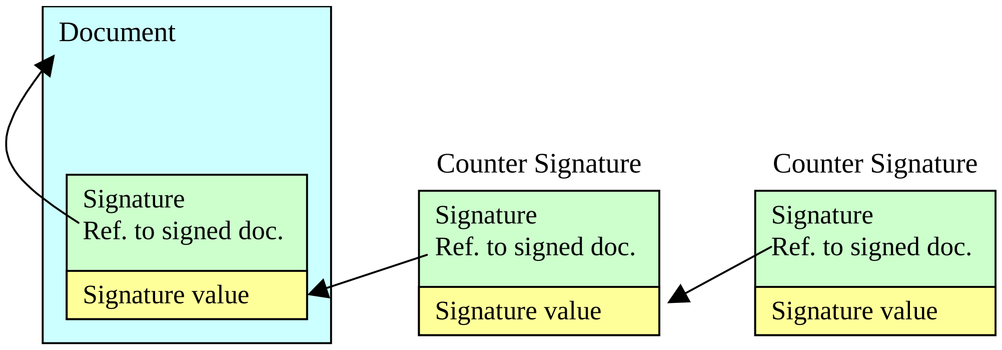

Figure 4: Embedded signatures

1.2. ES-T (Electronic Signature with Timestamp)

An ES-T is a signature to which a time-stamp token obtained from a TSA has been added, and this is associated with the signature value (SignatureValue in SignerInfo as defined in CMS) in the electronically signed document in order to provide a trusted time for the digital signature. The signature value is computed based on the hash value of the electronic document, and the time-stamp token generated from the signature value, together with the signature's time of existence, acts to provide a trusted time for the electronic data.

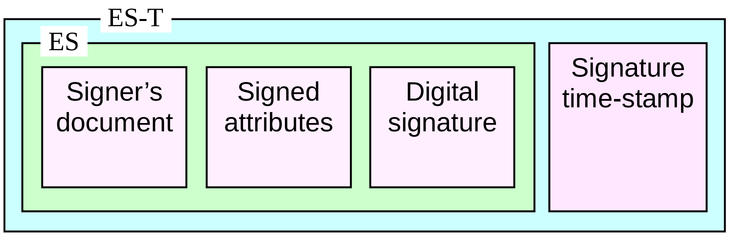

Figure 5: ES-T

Time-stamp tokens obtained from several different TSA's, but corresponding to the one signature, may be present. When several signatures are attached, time-stamp tokens may be obtained for each individual signature value, or the time-stamp tokens obtained may correspond only to one signature.

The Signature Timestamp attribute's OID may take the following values:

```
    id-aa-signatureTimeStampToken OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                  rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                                  id-aa(2) 14}
```

The Signature Timestamp attribute value has ASN.1 type SignatureTimeStampToken:

```
    SignatureTimeStampToken ::= TimeStampToken
```

The value of the messageImprint field within TimeStampToken shall be the value of the signature field (without the type or length encoding for that value) within SignerInfo for the signedData.

TimeStampToken is defined in RFC3161.

Validation data for SignatureTimestamp (certification path and revocation data, conforming to the ES validation data) may be encapsulated in one of the following:

1) certificates and crls within the time-stamp token
2) in the same place as the ES validation data (complete validation reference data and extended validation data)
3) in the unsigned attributes within the time-stamp token (in extended validation data form)

* 1) and 3) are recommended for build time, and all options, 1) - 3), must be handled for verification.

1.3. ES-C (Complete validation reference data)

ES-C builds on ES-T by adding reference information for all of the public key certificates in the certification path used in the verification of the digital signature (excluding revocation data for the signer's public key certificate), and for revocation data (including revocation data for the signer's public key certificate) for each of the public key certificates, such as the CRL, OCSP response, and so forth.

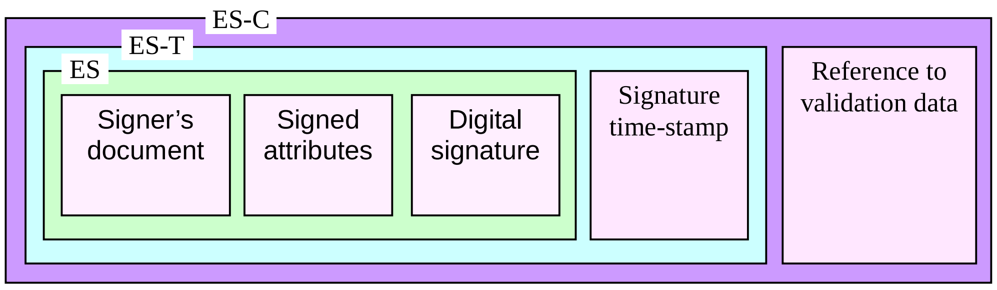

Figure 6: ES-C

An electronic signature with complete validation data references (ES-C) is an electronic signature document for which all the additional data required for validation (all certificates and revocation data) is available.

As a minimum the complete validation reference data shall contain:

- A Signature Timestamp attribute
- Complete Certificate Refs
- Complete Revocation Refs

The complete validation reference data may take the form of X-Long validation data containing the following information (in preparation for future situations where the verification process is unable to access this data)

- Complete Certificate Values **(MANDATORY)**
- Complete Revocation Values **(MANDATORY)**

The complete validation reference data may also take the form of extended validation data containing the following information (to protect against a future compromise of a CA, and to ensure the integrity of the validation data):

- ES-C Timestamp (present if ES-X Type 1 is used) **not used (may be ignored)**
- Time-Stamped Certificates and CRLs references (present if ES-X Type 2 is used) **not used (may be ignored)**

(i) Complete Certificate Refs attribute definition

The Complete Certificate Refs attribute is an unsigned attribute. Complete Certificate Refs references all CA certificates right up to the signer's certificate used in the verification of the ES (however, it does not include a reference to the signer's certificate).

Only one single instance of this attribute shall occur for any one signature.

Note 1: The signer certificate is referenced by the signing certificate attribute.

Note 2: The certification path of the signature time-stamp may be included.

The Complete Certificate Refs attribute is identified by the following OID:

```
    id-aa-ets-certificateRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                   rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                                   id-aa(2) 21}
```

The Complete Certificate Refs attribute value has the ASN.1 syntax CompleteCertificateRefs.

```
    CompleteCertificateRefs ::= SEQUENCE OF OtherCertID
```

IssuerSerial must be included in OtherCertID. certHash must match the certificate hash value referenced.

(ii) Complete Revocation Refs attribute definition

Complete Revocation Refs is an unsigned attribute. Only one single instance of this attribute shall occur for any one signature. This attribute references all CRLs, or OCSP responses for the signer and CA certificates required for verification of the ES-C.

Note: Revocation data for the certification path of the signature time-stamp may be included.

The Complete Revocation Refs attribute is identified by the following OID:

```
id-aa-ets-revocationRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                              rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                              id-aa(2) 22}
```

The Complete Revocation Refs attribute value has the ASN.1 syntax CompleteRevocationRefs.

```
CompleteRevocationRefs ::= SEQUENCE OF CrlOcspRef

CrlOcspRef ::= SEQUENCE {
         crlids         [0] CRLListID       OPTIONAL,
         ocspids        [1] OcspListID      OPTIONAL,
         otherRev       [2] OtherRevRefs    OPTIONAL
   }
```

CompleteRevocationRefs shall contain one CrlOcspRef for the signing-certificate, followed by one for each OtherCertID in the CompleteCertificateRefs attribute. The second and subsequent CrlOcspRef fields shall be in the same order as the OtherCertID to which they relate. At least one of CRLListID or OcspListID or OtherRevRefs should be present for all but the "trusted" CA of the certificate path. Revocation data other than CRLs and OCSP responses is not used.

```
CRLListID ::=  SEQUENCE {
         crls       SEQUENCE OF CrlValidatedID}

CrlValidatedID ::=  SEQUENCE {
         crlHash                OtherHash,
         crlIdentifier           CrlIdentifier OPTIONAL}

CrlIdentifier ::= SEQUENCE {
         crlissuer              Name,
         crlIssuedTime           UTCTime,
         crlNumber              INTEGER OPTIONAL
   }

OcspListID ::=  SEQUENCE {
         ocspResponses       SEQUENCE OF OcspResponsesID}

OcspResponsesID ::=  SEQUENCE {
         ocspIdentifier           OcspIdentifier,
         ocspRepHash              OtherHash   OPTIONAL
   }

OcspIdentifier ::= SEQUENCE {
         ocspResponderID   ResponderID,
           -- As in OCSP response data
         producedAt     GeneralizedTime
           -- As in OCSP response data
   }
```

When creating a crlValidatedID, the crlHash is computed over the entire DER encoded CRL including the signature. The crlIdentifier would normally be present unless the CRL can be inferred from other information.

The crlIdentifier is to identify the CRL using the issuer name and the CRL issued time, (the time thisUpdate contained in the issued CRL).

The crlListID attribute is an unsigned attribute. In the case that the identified CRL is a Delta CRL then references to the set of CRLs to provide a complete revocation list shall be included.

The OcspIdentifier is to identify the OCSP response using the issuer name and the time of issue of the OCSP response (the time producedAt contained in the issued OCSP response). To distinguish between two OCSP responses received at the same time, the hash of the response contained in the OcspResponsesID is used.

Note 1: Revocation data for the signature time-stamp is not included.

```
OtherRevRefs ::= SEQUENCE {
          otherRevRefType     OtherRevRefType
          otherRevRefs        ANY DEFINED BY otherRevRefType
   }

OtherRevRefType ::= OBJECT IDENTIFIER
```

1.4. Extended Validation Data (ES-X)

ES-X extends ES-C in order to protect against the future compromise of a CA, to ensure integrity of validation data, and to protect against future situations in which obtaining the data is difficult.

There are three ES-X formats, ES-X Long (Figure 7), ES-X Type 1 (Figure 8), ES-X Type 2 (Figure 9). This profile **recognizes ES-X Long only**.

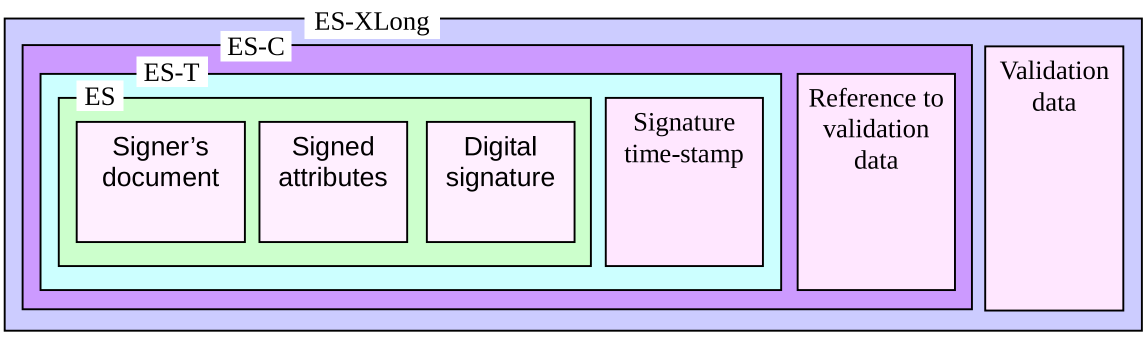

Figure 7: ES-X Long

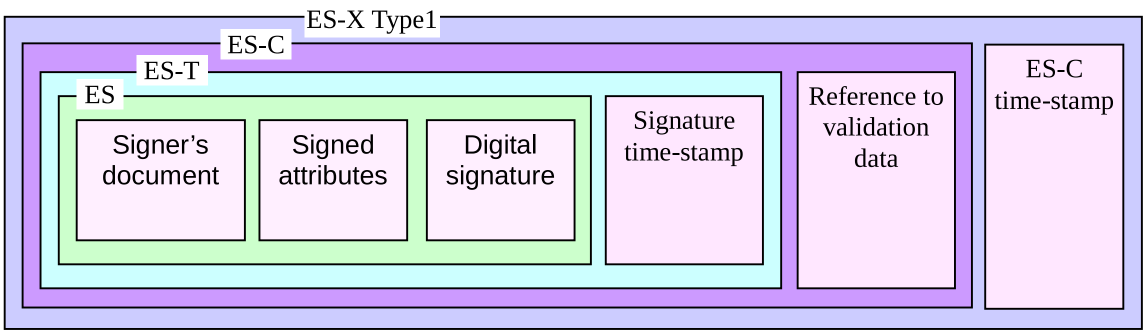

Figure 8: ES-X Type 1 **(not used)**

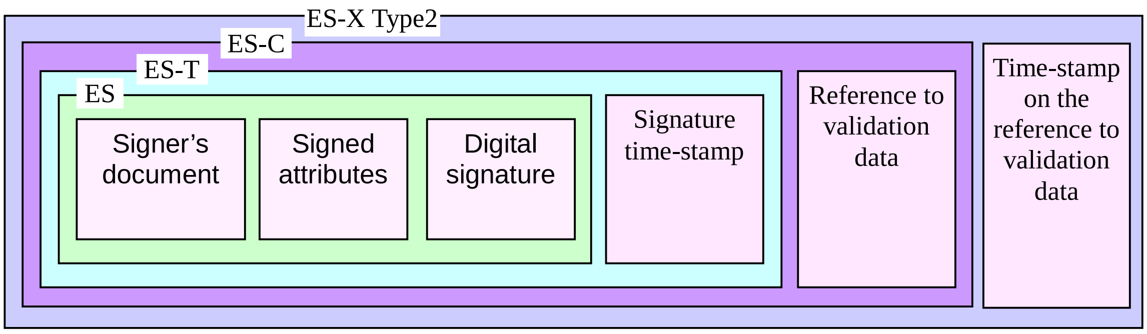

Figure 9: Type 2 **<u>(not used)</u>**

ES-X Long encapsulates the validation data within the electronic signature document. ES-X Type 1 adds a time-stamp on the ES-C itself, and ES-X Type 2 acquires and adds a time-stamp only on the references to the validation data.

Hash values for the validation data are included in the references to the validation data, however, the time-stamp is on the references, so if the hashing algorithms used later become vulnerable, then they become unable to certify that the references and validation data correspond to each other. In addition, to protect against the loss of the validation data itself (in particular, public key certificates of intermediate sub-CAs and revocation data, etc.), the validation data itself must also be retained.

For the long-term storage of digital signatures to be effective, the electronic document, the digital signature, the time-stamp and all of the validation data must be protected by a time-stamp, and by a tamper resistant framework. For long-term signature formats, this is achieved with the archive time-stamp described below. Basically, by adding an archive time-stamp at an appropriate time, the ES-C time-stamp and time-stamp on the validation data references are no longer required, and only protecting the actual validation data itself is important.

The ES-X Long format encapsulates all of the protected data. In order to implement systems that enable the long-term storage of electronically signed documents, it is sufficient if only ES-X Long is supported.

(i) Certificate Values attribute definition

Certificate Values is an unsigned attribute. Only one single instance of this attribute shall occur for any one signature. This attribute holds the certificates referenced by CompleteCertificationRefs, and the **<u>signer's certificate</u>**. **<u>(The signing certificate is included here since there are no other places specified where its inclusion is mandatory, and it is therefore not covered by archive time-stamps in SignedData Certificates and therefore not protected.)</u>**

Note: If an attribute certificate is used, it is not provided in this structure but provided as a signer-attributes attribute.

The Certificate Values attribute is identified by the following OID:

```
   id-aa-ets-certValues OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                              rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                              id-aa(2) 23}
```

The certificate-values attribute value has the ASN.1 syntax CertificateValues.

```
   CertificateValues ::=  SEQUENCE OF Certificate
```

Certificate is defined in RFC3280 and ITU-T Recommendation X.509.

(ii) Revocation Values attribute definition

The revocation-values attribute is an unsigned attribute. Only a single instance of this attribute shall occur with an electronic signature. It holds the values of CRLs and OCSP referenced in the CompleteRevocationRefs attribute.

The Revocation Values attribute is identified by the following OID:

```
    id-aa-ets-revocationValues OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                     rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                                     id-aa(2) 24}
```

The revocation-values attribute value has the ASN.1 syntax RevocationValues.

```
    RevocationValues ::=  SEQUENCE {
             crlVals          [0] SEQUENCE OF CertificateList    OPTIONAL,
             ocspVals         [1] SEQUENCE OF BasicOCSPResponse  OPTIONAL,
             otherRevVals     [2] OtherRevVals                  OPTIONAL ← not
             used
    }

    OtherRevVals ::= SEQUENCE {
             otherRevValType      OtherRevValType,
             otherRevVals         ANY DEFINED BY otherRevValType
    }

    OtherRevValType ::= OBJECT IDENTIFIER
```

Other revocation values **(not used.)**

CertificateList is defined in RFC3280 and ITU-T Recommendation X.509.

BasicOCSPResponse is defined in RFC2560.

(iii) ES-C Time-Stamp attribute definition **(not used; may be ignored.)**

(iv) Time-Stamped Certificates and CRLs attribute definitions **(not used; may be ignored.)**

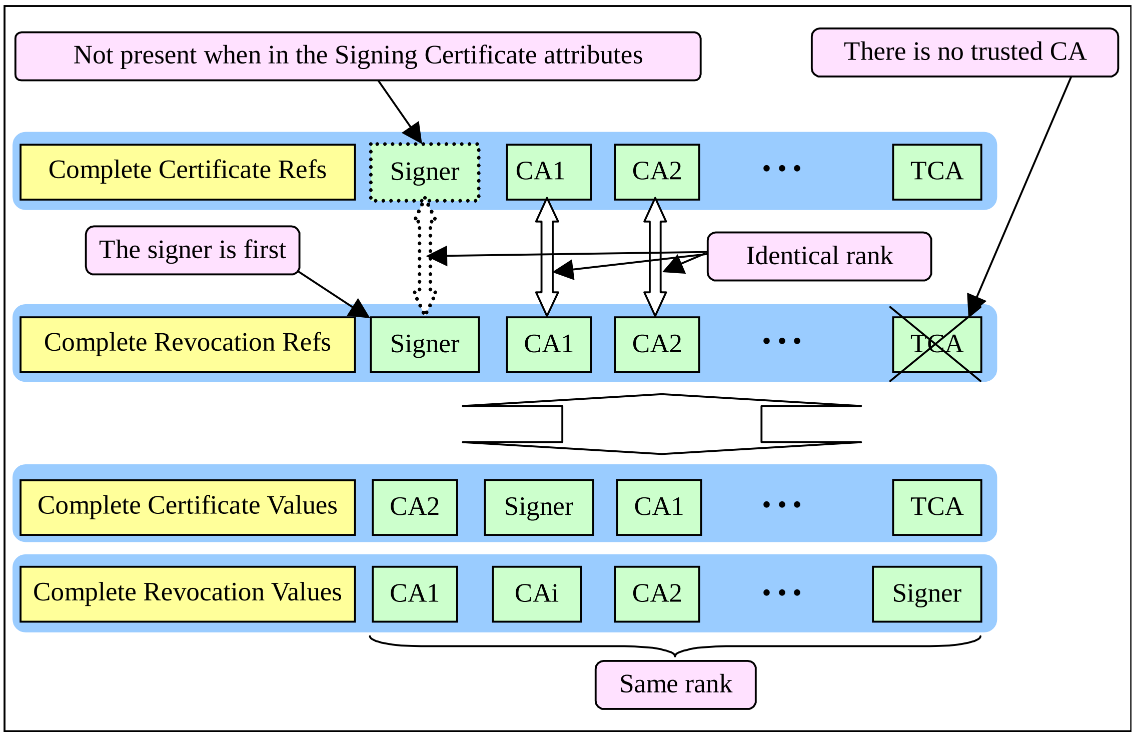

Figure 10: Relationship between Complete Validation Reference Data and Validation Values

1.5. ES-A (Archive Validation Data)

When attempting to extend the period of validity of an electronic signature to cover very long time frames, time-stamp signatures may be compromised, or TSA certificates may expire, and so several successive time-stamp signatures may be required. The archive time-stamp attribute is used for this. This time-stamp may be repeatedly applied over a period of time.

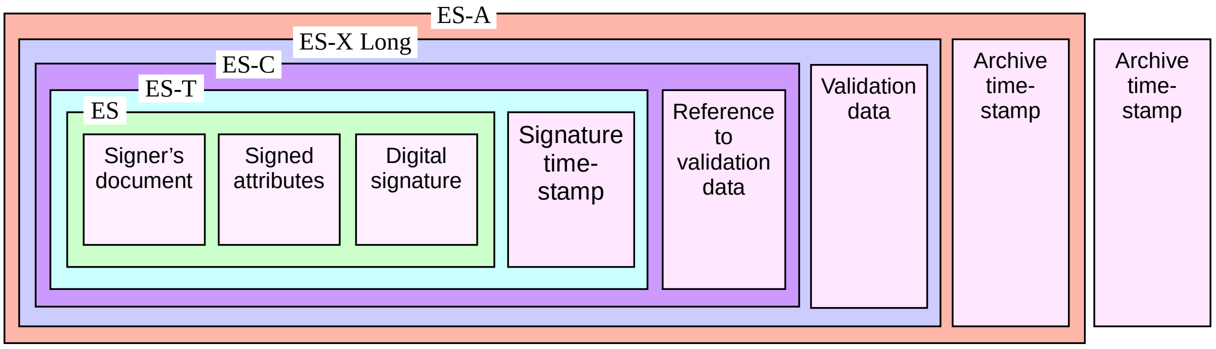

Figure 11: ES-A

(i) Archive Time-Stamp attribute definition

The Archive Time-Stamp is a time-stamp over the signer's document and the whole signature. If Certificate Values and Revocation Values attributes are not present, then these attributes must be added before a time-stamp is acquired. Archive Time-Stamp is an unsigned attribute. Several instances of this attribute may occur with an electronic signature both over time and from different TSAs.

The Archive Time-Stamp attribute is identified by the following OID:

```
   id-aa-ets-archiveTimestamp OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
                                  rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
                                  id-aa(2) 27}
```

Archive-time-stamp attribute values have the ASN.1 syntax ArchiveTimeStampToken.

```
   ArchiveTimeStampToken ::= TimeStampToken
```

The value of the messageImprint field within TimeStampToken shall be a hash of the concatenated values (without the type or length encoding for that value) of the following data objects in this order (refer to Figure 12):

- encapContentInfo eContent OCTET STRING;
- signedAttributes;
- signature field within SignerInfo;
- SignatureTimeStampToken attribute;
- CompleteCertificateRefs attribute;
- CompleteRevocationRefs attribute;
- CertificateValues attribute
  (If not already present this information shall be included in the ES-A.)
- RevocationValues attribute
  (If not already present this information shall be included in the ES-A.)
- ESCTimeStampToken attribute if present; **(not used (may not be covered by the time-stamp))**

- TimestampedCertsCRLs attribute if present; **<u>(not used (may not be covered by the time-stamp))</u>**
- any previous ArchiveTimeStampToken attributes.

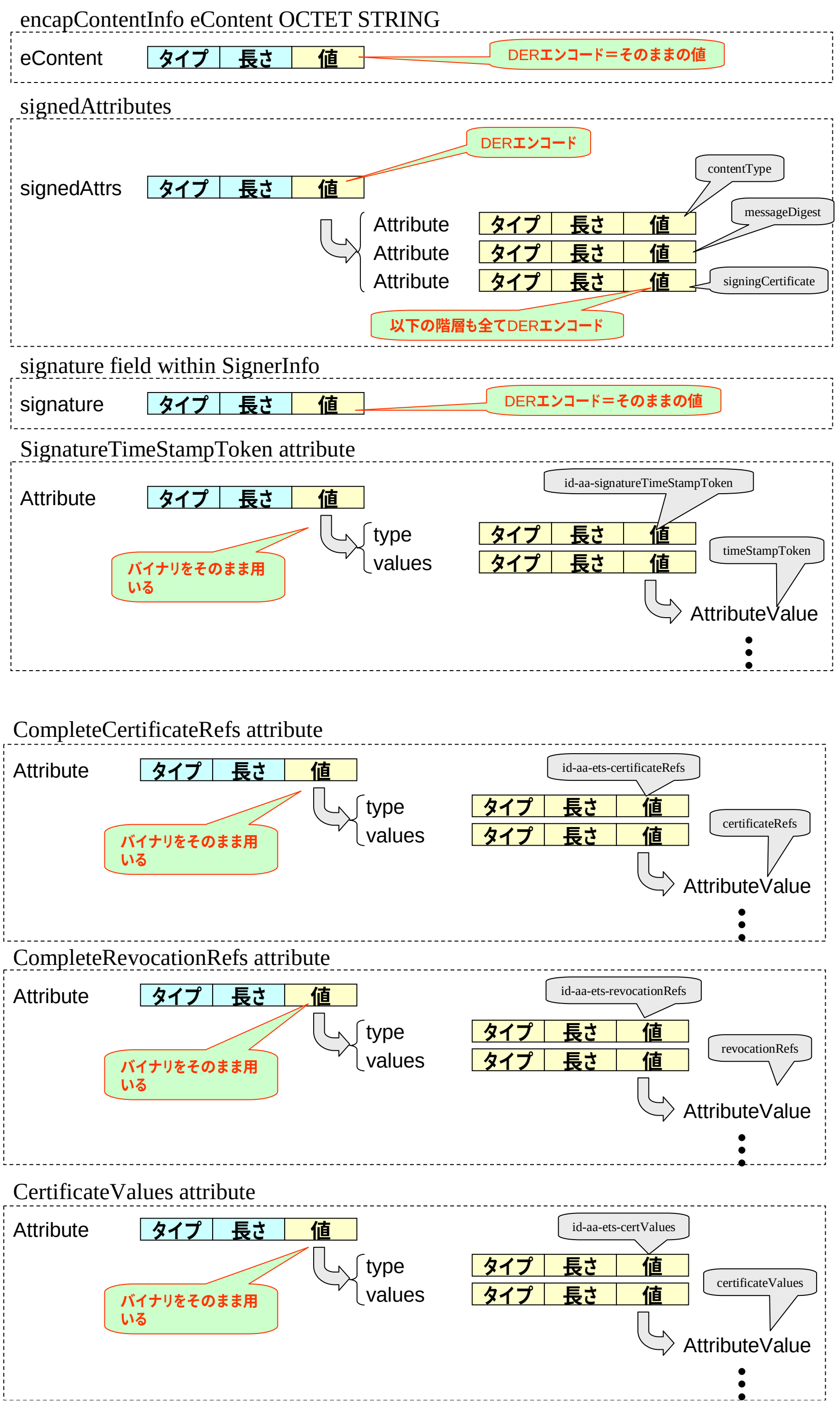
encapContentInfo eContent OCTET STRING
eContent
タイプ
長さ
値
DERエンコード＝そのままの値
signedAttributes
signedAttrs
DERエンコード
contentType
messageDigest
signingCertificate
Attribute
Attribute
Attribute
以下の階層も全てDERエンコード
signature field within SignerInfo
signature
DERエンコード＝そのままの値
SignatureTimeStampToken attribute
Attribute
バイナリをそのまま用いる
type
values
id-aa-signatureTimeStampToken
timeStampToken
AttributeValue
CompleteCertificateRefs attribute
id-aa-ets-certificateRefs
certificateRefs
CompleteRevocationRefs attribute
id-aa-ets-revocationRefs
revocationRefs
CertificateValues attribute
id-aa-ets-certValues
certificateValues

Figure 12: Items covered by the archive time-stamp (part I)

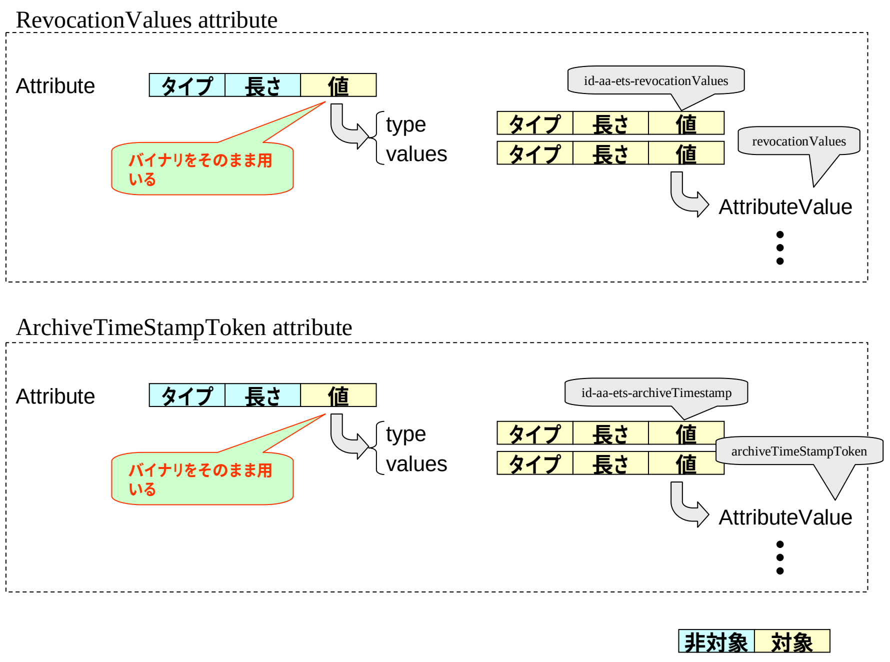

Figure 13: Items covered by the archive time-stamp (part II)

Refer to RFC3161 for information regarding TimeStampToken.

The time-stamp should be created using stronger algorithms (or longer key lengths) than in the original signatures.

Validation data for the archive time-stamp may be encapsulated in one of the following:

1) certificates and crls within the time-stamp token
2) in the unsigned attributes within the time-stamp token (in extended validation data form)

This profile recommends option 1) for build time, and both options 1) and 2) must be handled for verification.

The items obtained for the archive time-stamp in draft-pinkas-smime-cades-00.txt and ETSI TS 101 733 V1.5.1 are different. Basically, the value of the messageImprint field within TimeStampToken shall be a hash of the concatenated values (without the type or length encoding for that value) of the following data objects in this order:

- encapContentInfo in the SignedData
- Certificates and crls in the SignedData if they exist
- all SignerInfo data, including all signed and unsigned attributes

If the final item is to be covered, there are cases where the information covered by the archive time-stamp cannot be confirmed at the time of verification of each of the archive time-stamps. For example, this could occur if the countersignature was attached afterwards, or if an archive time-stamp was applied to a countersignature. Therefore, in this profile, previous specifications that clearly specify the target of time-stamps (RFC3126, TSI TS 101 733 V1.4.0 and prior versions) only are dealt with, and newer specifications are not handled.

1.6. Building long-term signatures

The following table shows when the various types of data should be obtained when building long-term signatures.

Format	Type of data	When it should be obtained
ES-T	Signature time-stamp	Before expiry or revocation of the signer's certificate. Should be obtained quickly following the generation or acquisition of the signature.
	Signature time-stamp validation data – all certificates in the certification path	Before acquisition of the 1st generation archive time-stamp
	Signature time-stamp validation data – revocation data for the above	Before acquisition of the 1st generation archive time-stamp
ES-C ES-XL	Signer's validation data – all certificates in the certification path	Before acquisition of the 1st generation archive time-stamp
	Signer's validation data – revocation data for the above	After the grace period following acquisition of the signature time-stamp
ES-A	1st generation archive time-stamp	Before the signature time-stamp and signer's validation data become invalid. (Before the expiry or revocation of the signature time-stamp certificate, and before the expiry or revocation of the certificate of the issuer (CA) of the signer's validation data.)
	1st generation archive time-stamp validation data – all certificates in the certification path	Before acquisition of the 2nd generation archive time-stamp
	1st generation archive time-stamp validation data – revocation data for the above	Before acquisition of the 2nd generation archive time-stamp
	2nd generation archive time –stamp	Before the expiry or revocation of the 1st generation archive time-stamp
	:	:
	nth generation archive time-stamp	Before the expiry or revocation of the (n-1)th generation archive time-stamp token
	nth generation archive time-stamp validation data – all certificates in the certification path	Before acquisition of the (n+1)th archive time-stamp
	nth generation archive time-stamp validation data – revocation data for the above	Immediately before acquisition of the (n+1)th archive time-stamp

The "grace period" when obtaining revocation data is the period between the issuance of the revocation request and the time when it is actually reflected in the revocation data (refer to Figure 14). The actual length of the grace period used is determined based on the policy of the certifying authority that issues the signer's certificate.

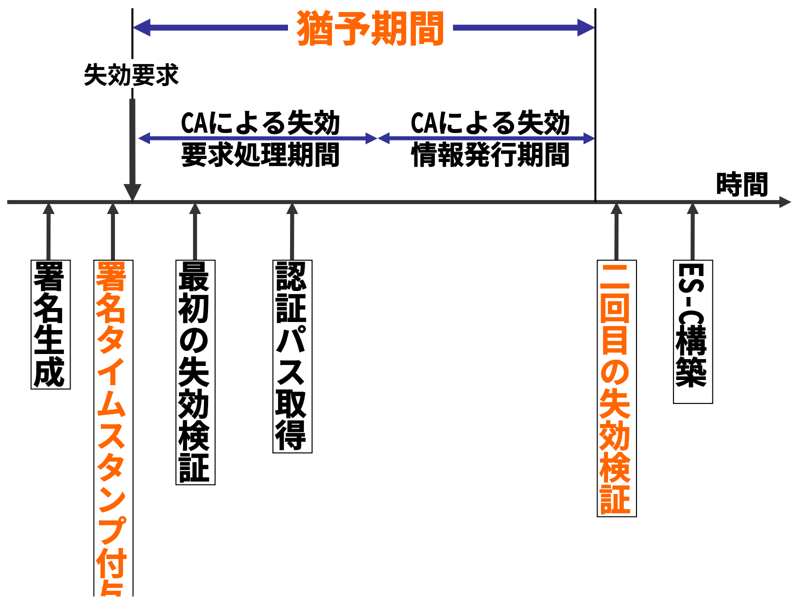

Figure 14: The grace period

1.7. Long-term signature verification

The following table shows the verification time for certificates.

Format	Type of certificate	When validity should be confirmed
ES-T	Signer's certificate	Time in the Signature Timestamp attribute's token
	Signature time-stamp TSA certificate	Current time
ES-C ES-XL	Signer's certificate	Time in the Signature Timestamp attribute's token
	Signature time-stamp TSA certificate	Current time, or when securely archived.
ES-A	Signer's certificate	Time in the Signature Timestamp attribute's token
	Signature time-stamp TSA certificate	Time in the 1st generation archive time-stamp's token
	1st generation archive time-stamp TSA certificate	Time in the 2nd generation archive time-stamp's token
	2nd generation archive time-stamp TSA certificate	Time in the 3rd generation archive time-stamp's token
	:	:
	(n-1)th generation archive time-stamp TSA certificate	Time in the nth generation archive time-stamp's token
	nth generation archive time-stamp TSA certificate	Current time

2. Summary of the CMS long-term signature profile

		CAdES BES	CAdES EPES	CAdES ES-T	CAdES ES-C	CAdES ES-X Long	CAdES ES-A
SignedAttributes		○	○	○	○	○	○
	ContentType	○	○	○	○	○	○
	MessageDigest	○	○	○	○	○	○
	SigningTime	△	△	△	△	△	△
	SigningCertificate	○	○	○	○	○	○
	SignaturePolicyIdentifier	×	○	△2	△2	△2	△2
	ContentReference	△	△	△	△	△	△
	ContentIdentifier	△	△	△	△	△	△
	ContentHints	△	△	△	△	△	△
	CommitmentTypeIndication	△	△	△	△	△	△
	SignerLocation	△	△	△	△	△	△
	SignerAttribute	△	△	△	△	△	△
	ContentTimeStamp	△	△	△	△	△	△
UnsignedAttribute		△	△	○	○	○	○
	CounterSignature	△	△	△	△	△	△
	SignatureTimeStamp	×	×	○	○	○	○
	CompleteCertificateRefs	×	×	×	○	○	○
	CompleteRevocationRefs	×	×	×	○	○	○
	AttributeCertificateRefs	×	×	×	△	△	△
	AttributeRevocationRefs	×	×	×	△	△	△
	CertificateValues	×	×	×	×	○	○
	RevocationValues	×	×	×	×	○	○
	ES-C TimeStamp	×	×	×	×	×	×
	TimeStampedCertsAndCrls	×	×	×	×	×	×
	ArchiveTimeStamp	×	×	×	×	×	○

○: Mandatory element

△: Optional element

△2: The SignaturePolicyIdentifier element is mandatory when EPES is used as a base

×: Not required (elements that must not be present)

Shaded cells: Conform with the standard specification

	ETSI TS 101 733 V 1.4.0 and prior versions (RFC3126)	ETSI TS 101 733 V 1.5.1
Coverage of the archive time-stamp calculation	Yes	--- *1

*1: This version includes some indeterminate elements in the calculation method, and is therefore not covered by the current version of this profile.