Network Working Group
Request for Comments: 3126
Category: Informational

D. Pinkas
Integris
J. Ross
N. Pope
Security & Standards

September 2001

Electronic Signature Formats for long term electronic signatures

Status of this Memo

Copyright Notice

Abstract

This document defines the format of an electronic signature that can remain valid over long periods. This includes evidence as to its validity even if the signer or verifying party later attempts to deny (i.e., repudiates the validity of the signature).

The format can be considered as an extension to RFC 2630 and RFC 2634, where, when appropriate additional signed and unsigned attributes have been defined.

The contents of this Informational RFC is technically equivalent to ETSI TS 101 733 V.1.2.2. The ETSI TS is under the ETSI Copyright (C). Individual copies of this ETSI deliverable can be downloaded from http://www.etsi.org

Table of Contents

1. Introduction

This document is intended to cover electronic signatures for various types of transactions, including business transactions (e.g., purchase requisition, contract, and invoice applications) where long term validity of such signatures is important. This includes evidence as to its validity even if the signer or verifying party later attempts to deny (i.e., repudiates, see [ISONR]) the validity of the signature).

Electronic signatures can be used for any transaction between an individual and a company, between two companies, between an individual and a governmental body, etc. This document is independent of any environment. It can be applied to any environment e.g., smart cards, GSM SIM cards, special programs for electronic signatures etc.

An electronic signature produced in accordance with this document provides evidence that can be processed to get confidence that some commitment has been explicitly endorsed under a signature policy, at a given time, by a signer under an identifier, e.g., a name or a pseudonym, and optionally a role.

The European Directive on a community framework for Electronic Signatures defines an electronic signature as: "data in electronic form which is attached to or logically associated with other electronic data and which serves as a method of authentication". An electronic signature as used in the current document is a form of advanced electronic signature as defined in the Directive.

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document (in uppercase, as shown) are to be interpreted as described in [RFC2119].

2 Overview

2.1 Aim

The aim of this document is to define an Electronic Signature (ES) that remains valid over long periods. This includes evidence as to its validity even if the signer or verifying party later attempts to deny (repudiates) the validity of the signature.

This document specifies the use of trusted service providers (e.g., Time-Stamping Authorities (TSA)), and the data that needs to be archived (e.g., cross certificates and revocation lists) to meet the requirements of long term electronic signatures. An electronic signature defined by this document can be used for arbitration in case of a dispute between the signer and verifier, which may occur at some later time, even years later. This document uses a signature policy, referenced by the signer, as the basis for establishing the validity of an electronic signature.

2.2 Basis of Present Document

This document is based on the use of public key cryptography to produce digital signatures, supported by public key certificates.

A Public key certificate is a public keys of a user, together with some other information, rendered unforgeable by encipherment with the private key of the Certification Authority (CA) which issued it (ITU-T Recommendation X.509 [1]).

This document also specifies the uses of time-stamping services to prove the validity of a signature long after the normal lifetime of critical elements of an electronic signature and to support non-repudiation. It also, as an option, defines the use of additional time-stamps to provide very long-term protection against key compromise or weakened algorithms.

This document builds on existing standards that are widely adopted. This includes:

* RFC 2459 [RFC2459] Internet X.509 Public Key Infrastructure Certificate and CRL Profile (PKIX);
* RFC 2630 [CMS] Crytographic Message Syntax (CMS);
* RFC 2634 [ESS] Enhanced Security Services (ESS);
* RFC 2439 [OCSP] One-line Certificate Status Protocol (OCSP);
* ITU-T Recommendation X.509 [1] Authentication framework;
* RFC (to be published) [TSP] PKIX Time Stamping protocol (TSP).

NOTE: See clause 8 for a full set of references.

2.3 Major Parties

The following are the major parties involved in a business transaction supported by electronic signatures as defined in this document:

* the Signer;
* the Verifier;
* the Arbitrator;
* Trusted Service Providers (TSP).

A Signer is an entity that initially creates the electronic signature. When the signer digitally signs over data using the prescribed format, this represents a commitment on behalf of the signing entity to the data being signed.

A verifier is an entity that verifies an evidence. (ISO/IEC 13888-1 [13]). Within the context of this document this is an entity that validates an electronic signature.
An arbitrator, is an entity which arbitrates disputes between a signer and a verifier when there is a disagreement on the validity of a digital signature.

Trusted Service Providers (TSPs) are one or more entities that help to build trust relationships between the signer and verifier. Use of some specific TSP services MAY be mandated by signature policy. TSP supporting services may provide the following information: user certificates, cross-certificates, time-stamping tokens, CRLs, ARLs, OCSP responses.

The following TSPs are used to support the validation or the verification of electronic signatures:

* Certification Authorities;
* Registration Authorities;
* Repository Authorities (e.g., a Directory);
* Time-Stamping Authorities;
* One-line Certificate Status Protocol responders;
* Attribute Authorities;
* Signature Policy Issuers.

Certification Authorities provide users with public key certificates.

Registration Authorities allows the registration of entities before a CA generates certificates.

Repository Authorities publish CRLs issued by CAs, cross-certificates (i.e., CA certificates) issued by CAs, signature policies issued by Signature Policy Issuers and optionally public key certificates (i.e., leaf certificates) issued by CAs.

Time-Stamping Authorities attest that some data was formed before a given trusted time.

One-line Certificate Status Protocol responders (OSCP responders) provide information about the status (i.e., revoked, not revoked, unknown) of a particular certificate.

A Signature Policy Issuer issues signatures policies that define the technical and procedural requirements for electronic signature creation, validation and verification, in order to meet a particular business need.

Attributes Authorities provide users with attributes linked to public key certificates

2.4 Electronic Signatures and Validation Data

Validation of an electronic signature in accordance with this document requires:

* The electronic signature; this includes:

  - the signature policy;
  - the signed user data;
  - the digital signature;
  - other signed attributes provided by the signer;
  - other unsigned attributes provided by the signer.

Validation data which is the additional data needed to validate the electronic signature; this includes:

  - certificates references;
  - certificates;
  - revocation status information references;
  - revocation status information;
  - time-stamps from Time Stamping Authorities (TSAs).

* The signature policy specifies the technical requirements on signature creation and validation in order to meet a particular business need. A given legal/contractual context may recognize a particular signature policy as meeting its requirements.

For example: a specific signature policy may be recognized by court of law as meeting the requirements of the European Directive for electronic commerce. A signature policy may be written using a formal notation like ASN.1 or in an informal free text form provided the rules of the policy are clearly identified. However, for a given signature policy there shall be one definitive form which has a unique binary encoded value.

Signed user data is the user's data that is signed.

The Digital Signature is the digital signature applied over the following attributes provided by the signer:

* hash of the user data (message digest);
* signature Policy Identifier;
* other signed attributes

The other signed attributes include any additional information which must be signed to conform to the signature policy or this document (e.g., signing time).

According to the requirements of a specific signature policy in use, various Validation Data shall be collected and attached to or associated with the signature structure by the signer and/or the verifier. The validation data includes CA certificates as well as revocation status information in the form of certificate revocation lists (CRLs) or certificate status information provided by an on-line service. Additional data also includes time-stamps and other time related data used to provide evidence of the timing of given events. It is required, as a minimum, that either the signer or verifier obtains a time-stamp over the signer's signature or a secure time record of the electronic signature must be maintained. Such secure records must not be undetectably modified and must record the time close to when the signature was first validated.

2.5 Forms of Validation Data

An electronic signature may exist in many forms including:

* the Electronic Signature (ES), which includes the digital signature and other basic information provided by the signer;

* the ES with Time-Stamp (ES-T), which adds a time-stamp to the Electronic Signature, to take initial steps towards providing long term validity;

* the ES with Complete validation data (ES-C), which adds to the ES-T references to the complete set of data supporting the validity of the electronic signature (i.e., revocation status information).

The signer must provide at least the ES form, but in some cases may decide to provide the ES-T form and in the extreme case could provide the ES-C form. If the signer does not provide ES-T, the verifier must either create the ES-T on first receipt of an electronic signature or shall keep a secure time record of the ES. Either of these two approaches provide independent evidence of the existence of the signature at the time it was first verified which should be near the time it was created, and so protects against later repudiation of the existence of the signature. If the signer does not provide ES-C the verifier must create the ES-C when the complete set of revocation and other validation data is available.

The ES satisfies the legal requirements for electronic signatures as defined in the European Directive on electronic signatures, see Annex C for further discussion on relationship of this document to the Directive. It provides basic authentication and integrity protection and can be created without accessing on-line (time-stamping) services. However, without the addition of a time-stamp or a secure time record the electronic signature does not protect against the threat that the signer later denies having created the electronic signature (i.e., does not provide non-repudiation of its existence).

The ES-T time-stamp or time record should be created close to the time that ES was created to provide protection against repudiation. At this time all the data needed to complete the validation may not be available but what information is readily available may be used to carry out some of the initial checks. For example, only part of the revocation information may be available for verification at that point in time. Generally, the ES-C form cannot be created at the same time as the ES, as it is necessary to allow time for any revocation information to be captured. Also, if a certificate is found to be temporarily suspended, it will be necessary to wait until the end of the suspension period.

The signer should only create the ES-C in situations where it was prepared to wait for a sufficient length of time after creating the ES form before dispatching the ES-C. This, however, has the advantage that the verifier can be presented with the complete set of data supporting the validity of the ES.

Support for ES-C by the verifier is mandated (see clause 6 for specific conformance requirements).

An Electronic Signature (ES), with the additional validation data forming the ES-T and ES-C is illustrated in Figure 1:

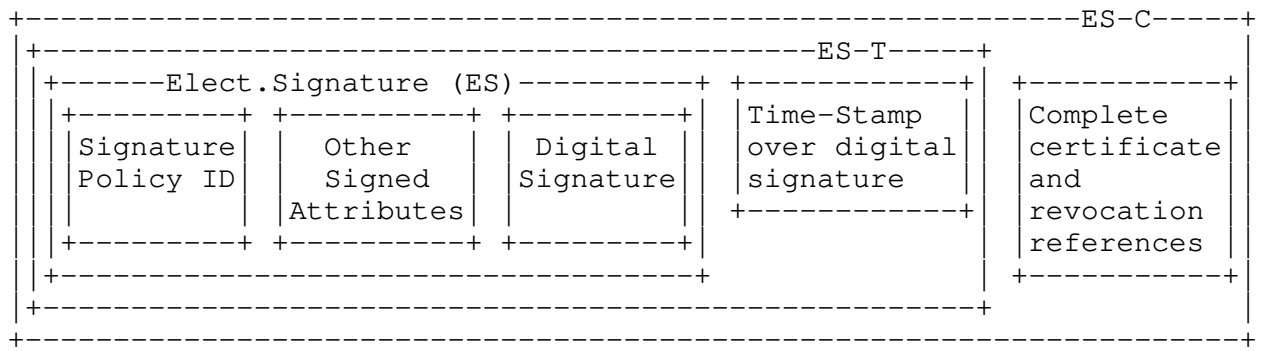

Figure 1: Illustration of an ES, ES-T and ES-C

The verifiers conformance requirements of an ES with a time-stamp of the digital signature is defined in subclause 6.2.

The ES on its own satisfies the legal requirements for electronic signatures as defined in the European Directive on electronic signatures. The signers conformance requirements of an ES are defined in subclause 6.1, and are met using a structure as indicated in figure 2:

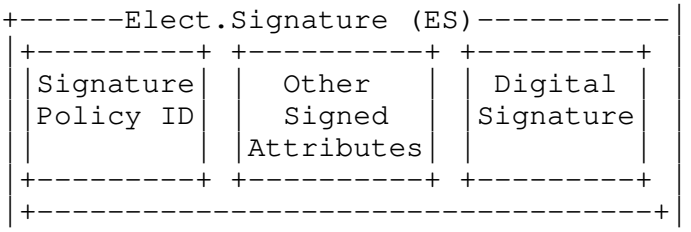

Figure 2: Illustration of an ES

Where there are requirements for long term signatures without time-stamping the digital signature, then a secure record is needed of the time of verification in association with the electronic signature (i.e., both must be securely recorded). In addition the certificates and revocation information used at the time of verification should to be recorded as indicated in figure 3 as an ES-C(bis).

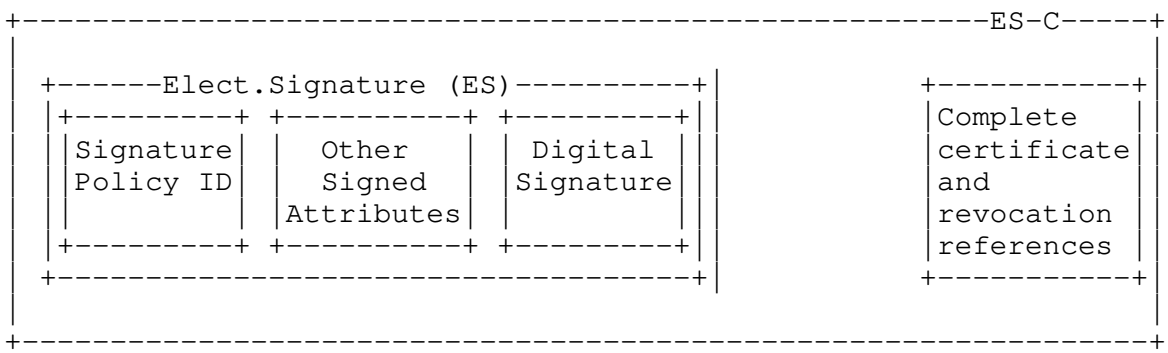

Figure 3: Illustration of an ES-C(bis)

The verifiers conformance requirements of an ES-C(bis) is defined in subclause 6.3.

Note: A time-stamp attached to the electronic signature or a secure time record helps to protect the validity of the signature even if some of the verification data associated with the signature become compromised AFTER the signature was generated. The time-stamp or a secure time record provides evidence that the signature was generated BEFORE the event of compromise; hence the signature will maintain its validity status.

2.6 Extended Forms of Validation Data

The complete validation data (ES-C) described above may be extended to form an ES with eXtended validation data (ES-X) to meet following additional requirements.

Firstly, when the verifier does not has access to,

* the signer’s certificate,
* all the CA certificates that make up the full certification path,
* all the associated revocation status information, as referenced in the ES-C.

then the values of these certificates and revocation information may be added to the ES-C. This form of extended validation data is called a X-Long.

Secondly, if there is a risk that any CA keys used in the certificate chain may be compromised, then it is necessary to additionally time-stamp the validation data by either:

* time-stamping all the validation data as held with the ES(ES-C), this eXtended validation data is called a Type 1 X-Time-Stamp; or
* time-stamping individual reference data as used for complete validation.

This form of eXtended validation data is called a Type 2 X-Time-Stamp.

NOTE: The advantages/drawbacks for Type 1 and Type 2 X-Time-Stamp are discussed in this document (see clause B.4.6.)

If all the above conditions occur then a combination of the two formats above may be used. This form of eXtended validation data is called a X-Long-Time-Stamped.

Support for the extended forms of validation data is optional.

An Electronic Signature (ES) , with the additional validation data forming the ES-X long is illustrated in Figure 4:

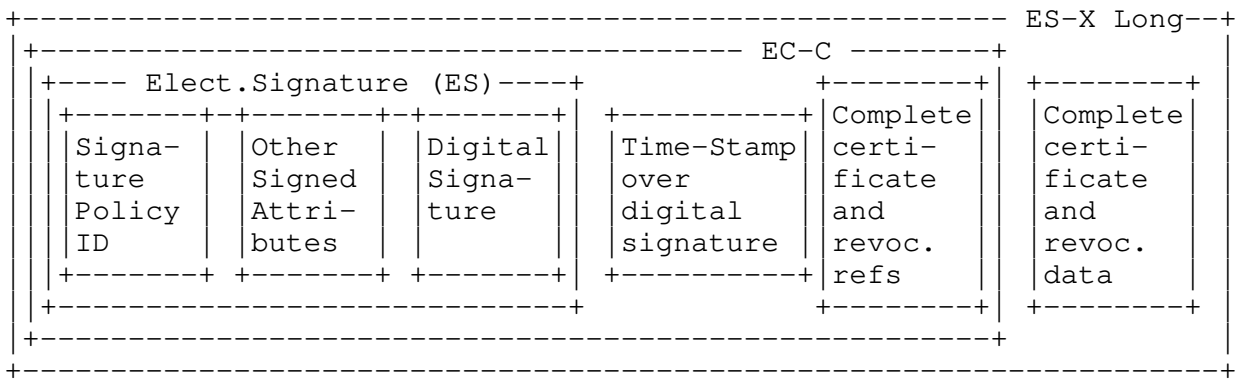

Figure 4: Illustration of an ES and ES-X long.

An Electronic Signature (ES) , with the additional validation data forming the eXtended Validation Data - Type 1 is illustrated in Figure 5:

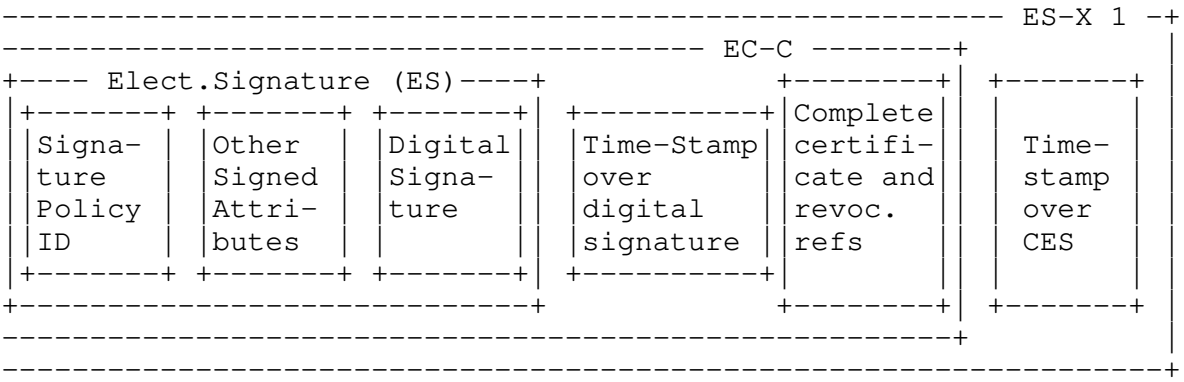

Figure 5: Illustration of ES with ES-X Type 1

An Electronic Signature (ES) , with the additional validation data forming the eXtended Validation Data - Type 2 is illustrated in Figure 6:

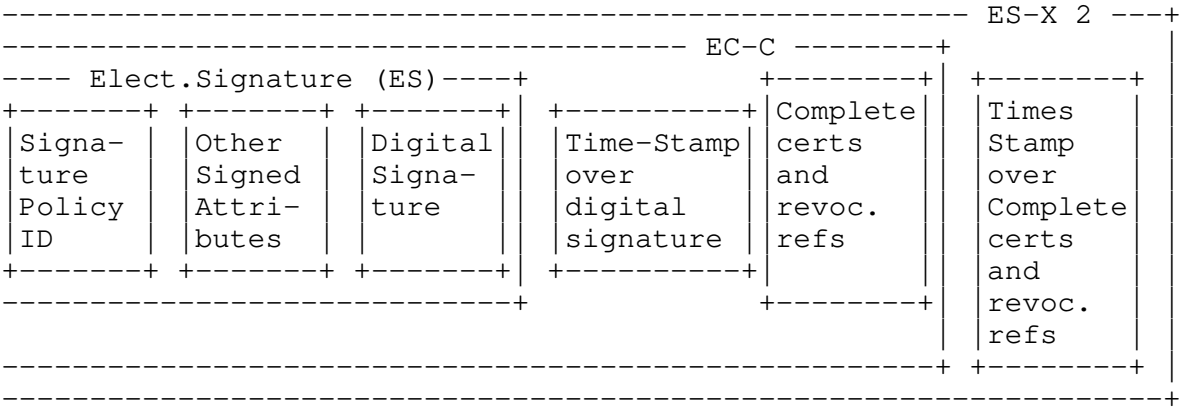

Figure 6: Illustration of ES with ES-X Type 2

2.7 Archive Validation Data

Before the algorithms, keys and other cryptographic data used at the time the ES-C was built become weak and the cryptographic functions become vulnerable, or the certificates supporting previous time-stamps expires, the signed data, the ES-C and any additional information (ES-X) should be time-stamped. If possible this should use stronger algorithms (or longer key lengths) than in the original time-stamp.

This additional data and time-stamp is called Archive Validation Data (ES-A). The Time-Stamping process may be repeated every time the protection used to time-stamp a previous ES-A become weak. An ES-A may thus bear multiple embedded time stamps.

An example of an Electronic Signature (ES), with the additional validation data for the ES-C and ES-X forming the ES-A is illustrated in Figure 7.

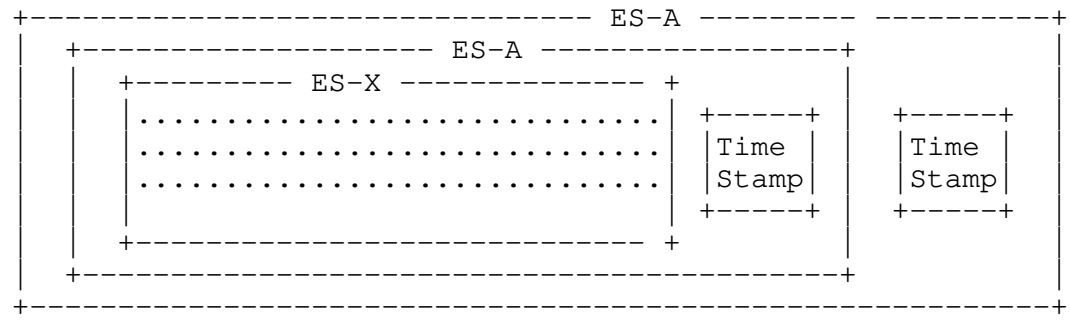

Figure 7: Illustration of ES -A

Support for ES-A is optional.

2.8 Arbitration

The ES-C may be used for arbitration should there be a dispute between the signer and verifier, provided that:

* a copy of the signature policy referenced by the signer is available;

* the arbitrator knows where to retrieve the signer's certificate (if not already present), all the cross-certificates and the required CRLs and/or OCSPs responses referenced in the ES-C;

* none of the issuing key from the certificate chain have ever been compromised;

* the cryptography used at the time the ES-C was built has not been broken at the time the arbitration is performed.

When the second condition is not met, then the plaintiff must provide an ES-X Long.

When it is known by some external means that the third condition is not met, then the plaintiff must provide an ES-X Time-Stamped.

When the two previous conditions are not met, the plaintiff must provide the two above information (i.e., an ES-X Time-Stamped and Long).

When the last condition is not met, the plaintiff must provide an ES-A.

It should be noticed that a verifier may need to get two time stamps at two different instants of time: one soon after the generation of the ES and one soon after some grace period allowing any entity from the certification chain to declare a key compromise.

2.9 Validation Process

The Validation Process validates an electronic signature in accordance with the requirements of the signature policy. The output status of the validation process can be:

* valid;
* invalid;
* incomplete verification.

A Valid response indicates that the signature has passed verification and it complies with the signature validation policy.

A signature validation policy is a part of the signature policy which specifies the technical requirements on the signer in creating a signature and verifier when validating a signature.

An Invalid response indicates that either the signature format is incorrect or that the digital signature value fails verification (e.g., the integrity checks on the digital signature value fails or any of the certificates on which the digital signature verification depends is known to be invalid or revoked).

An Incomplete Validation response indicates that the format and digital signature verifications have not failed but there is insufficient information to determine if the electronic signature is valid under the signature policy. This can include situations where additional information, which does not effect the validity of the digital signature value, may be available but is invalid.

In the case of Incomplete Validation, it may be possible to request that the electronic signature be checked again at a later date when additional validation information might become available. Also, in the case of incomplete validation, additional information may be made available to the application or user, thus allowing the application or user to decide what to do with partially correct electronic signatures.

The validation process may also output validation data:

* a signature time-stamp;
* the complete validation data;
* the archive validation data.

2.10 Example Validation Sequence

Figure 8, and subsequent description, describes how the validation process may build up a complete electronic signature over time.

Soon after receiving the electronic signature (ES) from the signer (1), the digital signature value may be checked, the validation process must at least add a time-stamp (2), unless the signer has provided one which is trusted by the verifier. The validation process may also validate the electronic signature, as required under the identified signature policy, using additional data (e.g., certificates, CRL, etc.) provided by trusted service providers. If the validation process is not complete then the output from this stage is the ES-T.

When all the additional data (e.g., the complete certificate and revocation information) necessary to validate the electronic signature first becomes available, then the validation process:

* obtains all the necessary additional certificate and revocation status information;

* completes all the validation checks on the ES, using the complete certificate and revocation information (if a time-stamp is not already present, this may be added at the same stage combining ES-T and ES-C process);

* records the complete certificate and revocation references (3);

* indicates the validity status to the user (4).

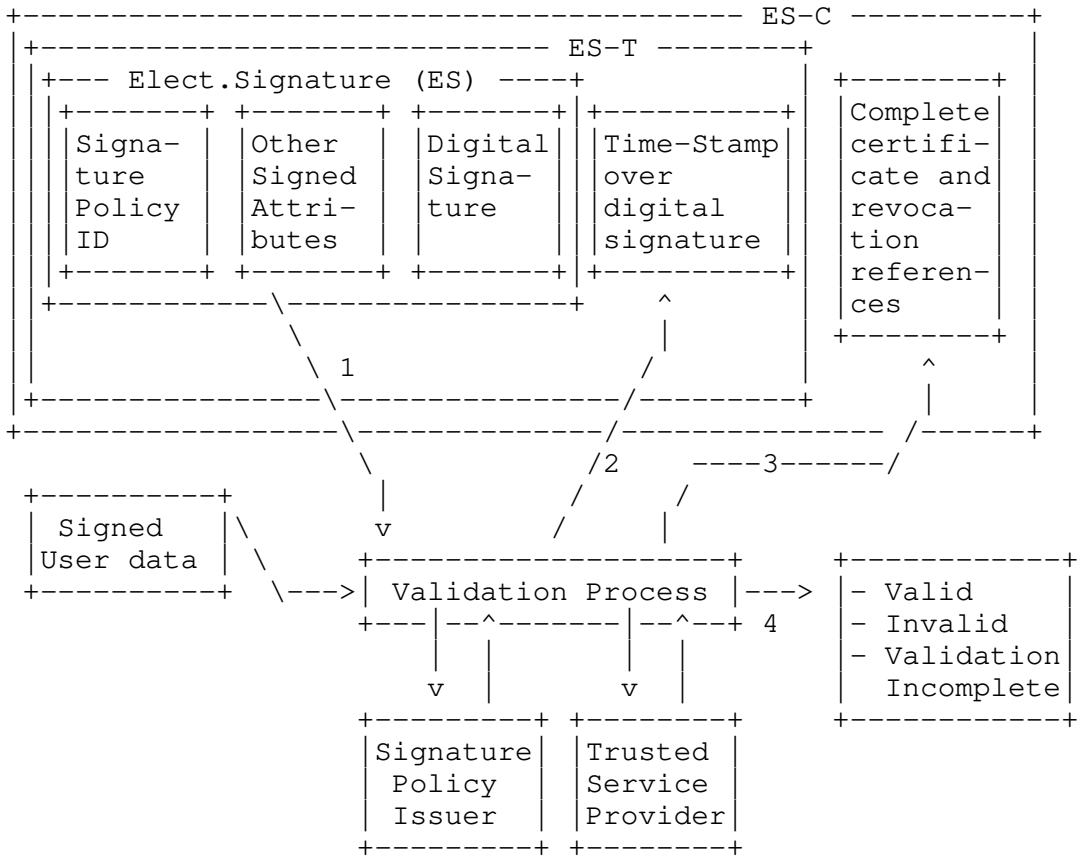

Figure 8: Illustration of an ES with Complete validation data (ES-C)

At the same time as the validation process creates the ES-C, the validation process may provide and/or record the values of certificates and revocation status information used in ES-C, called the ES-X Long (5). This is illustrated in figure 9:

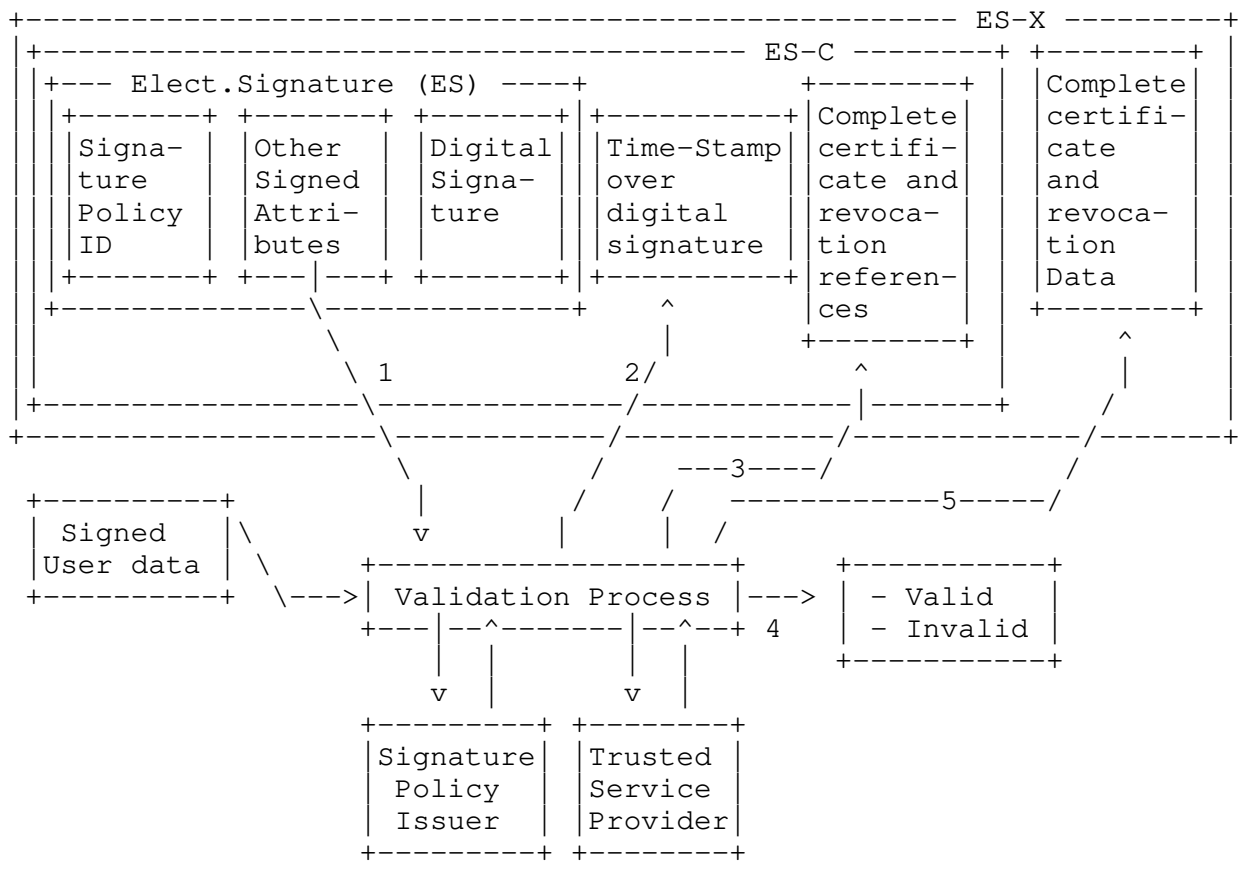

Figure 9: Illustration ES with eXtended validation data (Long)

When the validation process creates the ES-C it may also create extended forms of validation data. A first alternative is to time-stamp all data forming the Type 1 X-Time-Stamp (6). This is illustrated in figure 10:

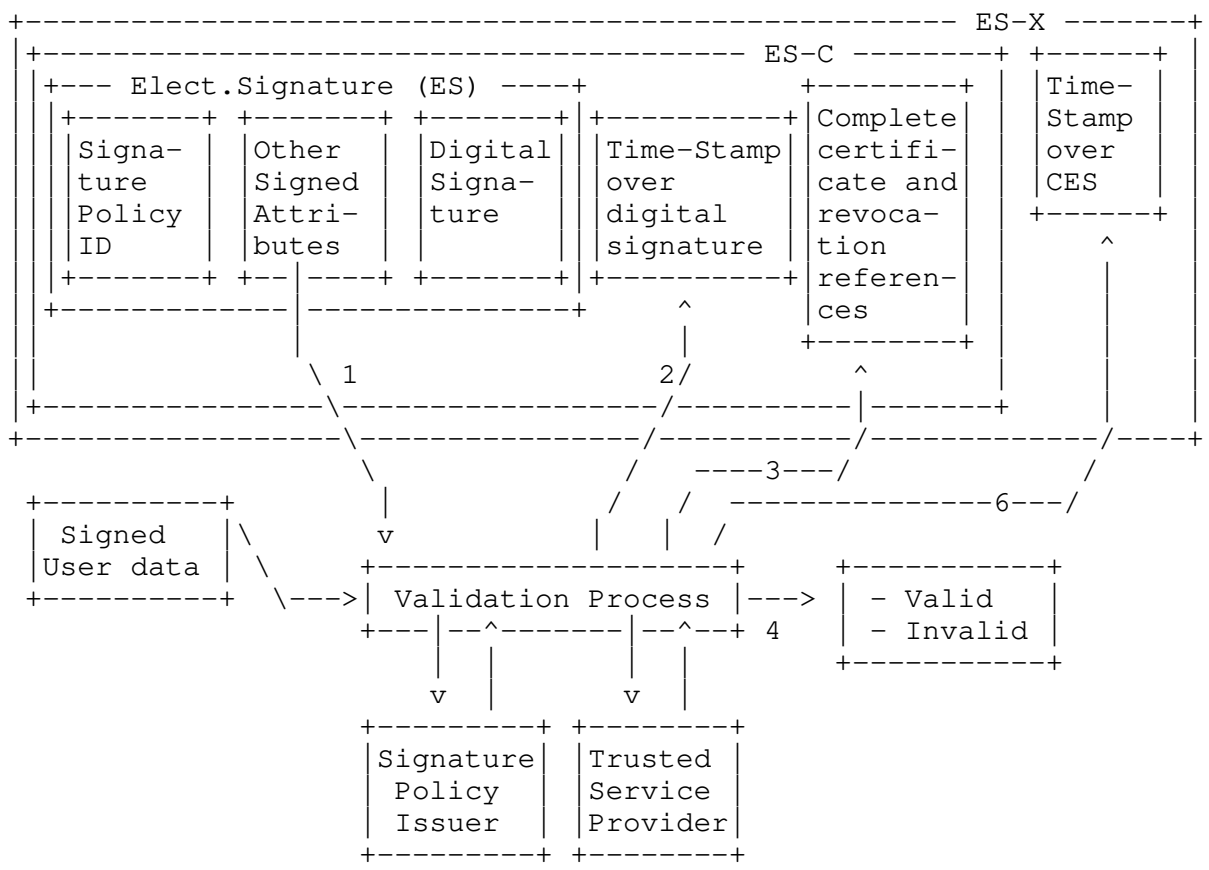

Figure 10: Illustration of ES with eXtended validation data - Type 1 X-Time-Stamp

Another alternative is to time-stamp the certificate and revocation information references used to validate the electronic signature (but not the signature) (6'); this is called Type 2 X-Time-Stamped. This is illustrated in figure 11:

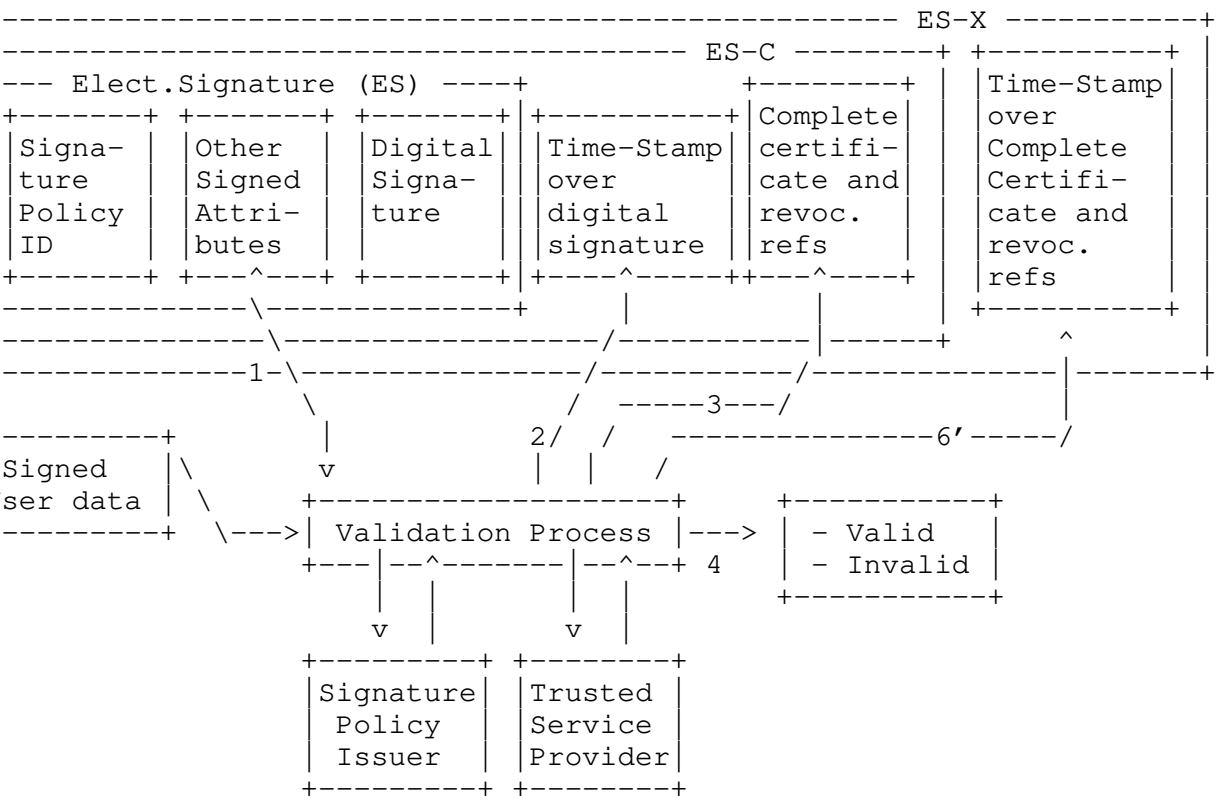

Figure 11: Illustration of ES with eXtended validation data - Type 2 X-Time-Stamp

Before the algorithms used in any of electronic signatures become or are likely, to be compromised or rendered vulnerable in the future, it is necessary to time-stamp the entire electronic signature, including all the values of the validation and user data as an ES with Archive validation data (ES-A)

An ES-A is illustrated in figure 12:

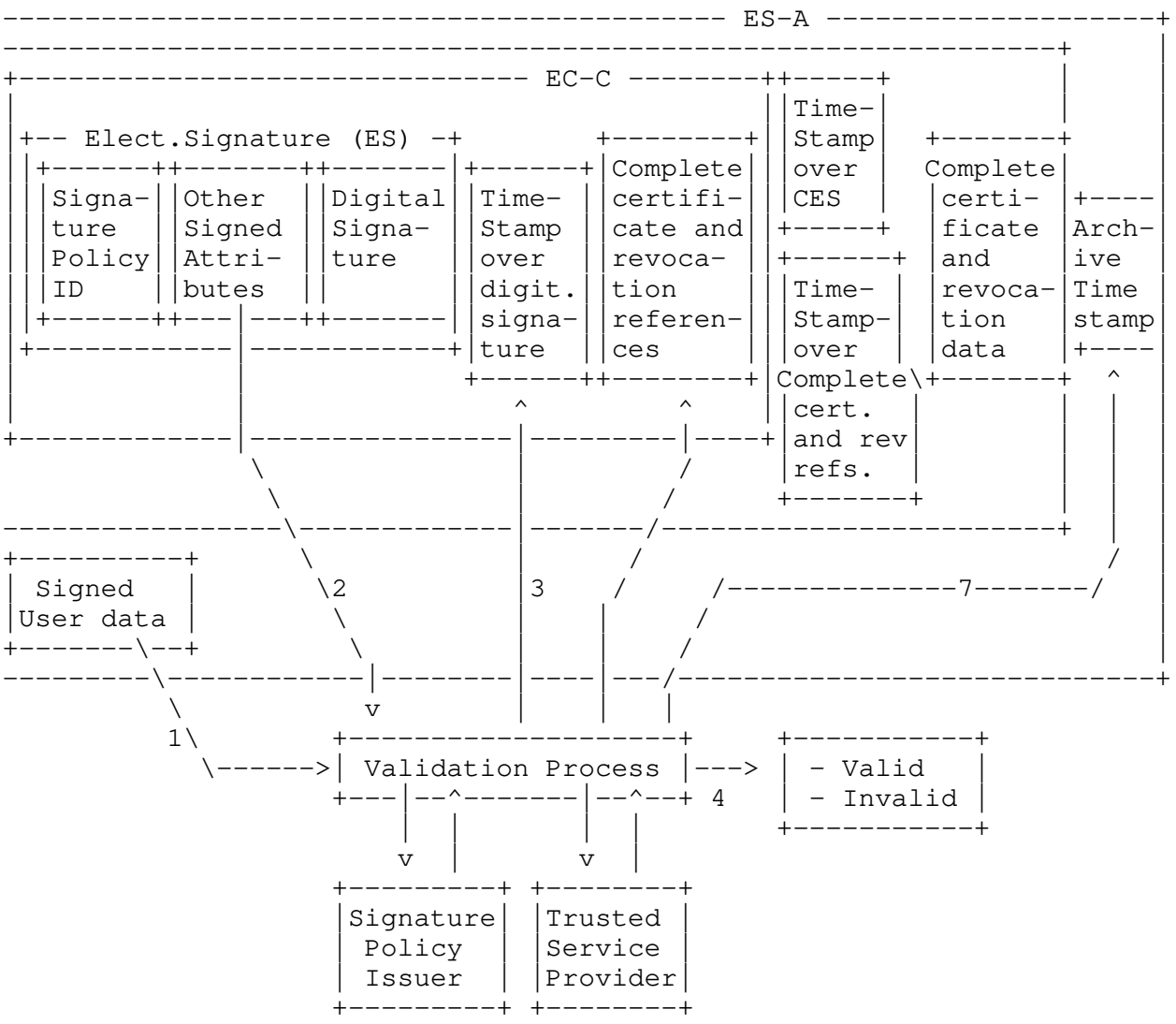

Figure 12: Illustration of an ES with Archive validation data (ES-A)

2.11 Additional optional features of an ES

This document also defines additional optional features of an electronic signature to:

* indicate a commitment type being made by the signer;
* indicate the role under which a signature was created;
* support multiple signatures.

3. Data structure of an Electronic Signature

This clause uses and builds upon the Cryptographic Message Syntax (CMS), as defined in RFC 2630 [CMS], and Enhanced Security Services (ESS), as defined in RFC 2634 [ESS]. The overall structure of Electronic Signature is as defined in [CMS]. The Electronic Signature (ES) uses attributes defined in [CMS], [ESS] and this document. This document defines in full the ES attributes which it uses and are not defined elsewhere.

The mandated set of attributes and the digital signature value is defined as the minimum Electronic Signature (ES) required by this document. A signature policy MAY mandate other signed attributes to be present.

3.1 General Syntax

The general syntax of the ES is as defined in [CMS].

3.2 Data Content Type

The data content type of the ES is as defined in [CMS].

The data content type is intended to refer to arbitrary octet strings, such as ASCII text files; the interpretation is left to the application. Such strings need not have any internal structure (although they could have their own ASN.1 definition or other structure).

3.3 Signed-data Content Type

The Signed-data content type of the ES is as defined in [CMS].

The signed-data content type consists of a content of any type and zero or more signature values. Any number of signers in parallel can sign any type of content. The typical application of the signed-data content type represents one signer's digital signature on content of the data content type.

To make sure that the verifier uses the right certificate, this document mandates that the hash of the signers certificate is always included in the Signing Certificate signed attribute.

3.4 SignedData Type

The syntax of the SignedData type of the ES is as defined in [CMS].

The fields of type SignedData have the meanings defined [CMS] except that:

* version is the syntax version number. The value of version must be 3.

* The identification of signer's certificate used to create the signature is always present as a signed attribute.

* The degenerate case where there are no signers is not valid in this document.

3.5 EncapsulatedContentInfo Type

The syntax of the EncapsulatedContentInfo a type of the ES is as defined in [CMS].

For the purpose of long term validation as defined by this document, it is advisable that either the eContent is present, or the data which is signed is archived in such as way as to preserve the any data encoding. It is important that the OCTET STRING used to generate the signature remains the same every time either the verifier or an arbitrator validates the signature.

The degenerate case where there are no signers is not valid in this document.

3.6 SignerInfo Type

The syntax of the SignerInfo a type of the ES is as defined in [CMS].

Per-signer information is represented in the type SignerInfo. In the case of multiple independent signatures, there is an instance of this field for each signer.

The fields of type SignerInfo have the meanings defined in [CMS] except that signedAttributes must, as a minimum, contain the following attributes:

* ContentType as defined in clause 3.7.1.
* MessageDigest as defined in clause 3.7.2.
* SigningTime as defined in clause 3.7.3.
* SigningCertificate as defined in clause 3.8.1.
* SignaturePolicyId as defined in clause 3.9.1.

3.6.1 Message Digest Calculation Process

The message digest calculation process is as defined in [CMS].

3.6.2 Message Signature Generation Process

The input to the digital signature generation process is as defined in [CMS].

3.6.3 Message Signature Verification Process

The procedures for CMS signed data validation are as defined in [CMS] and enhanced in this document.

The input to the signature verification process includes the signer's public key verified as correct using either the ESS Signing Certificate attribute or the Other Signing Certificate attribute.

3.7 CMS Imported Mandatory Present Attributes

The following attributes MUST be present with the signed-data defined by this document. The attributes are defined in [CMS].

3.7.1 Content Type

The syntax of the content-type attribute type of the ES is as defined in [CMS].

3.7.2 Message Digest

The syntax of the message-digest attribute type of the ES is as defined in [CMS].

3.7.3 Signing Time

The syntax of the message-digest attribute type of the ES is as defined in [CMS] and further qualified by this document.

The signing-time attribute type specifies the time at which the signer claims to have performed the signing process.

This present document recommends the use of GeneralizedTime.

3.8 Alternative Signing Certificate Attributes

One, and only one, of the following two alternative attributes MUST be present with the signed-data defined by this document to identify the signing certificate. Both attributes include an identifier and a hash of the signing certificate. The first, which is adopted in existing standards, may be only used with the SHA-1 hashing algorithm. The other shall be used when other hashing algorithms are to be supported.

The signing certificate attribute is designed to prevent the simple substitution and re-issue attacks, and to allow for a restricted set of authorization certificates to be used in verifying a signature.

3.8.1 ESS Signing Certificate Attribute Definition

The syntax of the signing certificate attribute type of the ES is as defined in [ESS], and further qualified and profile in this document.

The ESS signing certificate attribute must be a signed attribute.

This document mandates the presence of this attribute as a signed CMS attribute, and the sequence must not be empty. The certificate used to verify the signature must be identified in the sequence, the Signature Validation Policy may mandate other certificate references to be present, that may include all the certificates up to the point of trust. The encoding of the ESSCertID for this certificate must include the issuerSerial field.

The issuerAndSerialNumber present in the SignerInfo must be consistent with issuerSerial field. The certificate identified must 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 must be considered invalid.

The sequence of policy information field is not used in this document.

NOTE: Where an attribute certificate is used by the signer to associate a role, or other attributes of the signer, with the electronic signature this is placed in the Signer Attribute attribute as defined in clause 3.12.3.

3.8.2 Other Signing Certificate Attribute Definition

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

This attribute must be used in the same manner as defined above for the ESS SigningCertificate attribute.

The following object identifier identifies the signing certificate attribute:

```
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 }
```

The signing certificate attribute value has the ASN.1 syntax OtherSigningCertificate

```
OtherSigningCertificate ::=  SEQUENCE {
    certs        SEQUENCE OF OtherCertID,
    policies     SEQUENCE OF PolicyInformation OPTIONAL
                 -- NOT USED IN THIS DOCUMENT
}

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

OtherHash ::= CHOICE {
    sha1Hash OtherHashValue,  -- This contains a SHA-1 hash
    otherHash OtherHashAlgAndValue
}

OtherHashValue ::= OCTET STRING

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

3.9 Additional Mandatory Attributes

3.9.1 Signature policy Identifier

This document mandates that a reference to the signature policy, is included in the signedData, this reference is either explicitly identified or implied by the semantics of the signed content and other external data. A signature policy defines the rules for creation and validation of an electronic signature, is included as a signed attribute with every signature. The signature policy identifier must be a signed attribute.

The following object identifier identifies the signature policy identifier 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 }
```

Signature-policy-identifier attribute values have ASN.1 type SignaturePolicyIdentifier.

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


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

SignaturePolicyImplied ::= NULL
```

The presence of the NULL type indicates that the signature policy is implied by the semantics of the signed data and other external data.

The sigPolicyId field contains an object-identifier which uniquely identifies a specific version of the signature policy. The syntax of this field is as follows:

```
   SigPolicyId ::= OBJECT IDENTIFIER
```

The sigPolicyHash field contains the identifier of the hash algorithm and the hash of the value of the signature policy.

If the signature policy is defined using a computer processable notation like ASN.1, then the hash is calculated on the value without the outer type and length fields and the hashing algorithm must be as specified in the field signPolicyHshAlg.

If the signature policy is defined using another structure, the type of structure and the hashing algorithm must be either specified as part of the signature policy, or indicated using a signature policy qualifier.

```
   SigPolicyHash ::= OtherHashAlgAndValue
```

A signature policy identifier may be qualified with other information about the qualifier. The semantics and syntax of the qualifier is as associated with the object-identifier in the sigPolicyQualifierId field. The general syntax of this qualifier is as follows:

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

This document specifies the following qualifiers:

* spuri: This contains the web URI or URL reference to the signature policy

* spUserNotice: This contains a user notice which should be displayed whenever the signature is validated.

```
-- sigpolicyQualifierIds defined in this document

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))
}
```

3.10 CMS Imported Optional Attributes

The following attributes MAY be present with the signed-data defined by this document. The attributes are defined in ref [CMS] and are imported into this specification and were appropriate qualified and profiling by this document.

3.10.1 Countersignature

The syntax of the countersignature attribute type of the ES is as defined in [CMS]. The countersignature attribute must be an unsigned attribute.

3.11 ESS Imported Optional Attributes

The following attributes MAY be present with the signed-data defined by this document. The attributes are defined in ref [ESS] and are imported into this specification and were appropriate qualified and profiling by this document.

3.11.1 Content Reference Attribute

The content reference attribute is a link from one SignedData to another. It may be used to link a reply to the original message to which it refers, or to incorporate by reference one SignedData into another.

The content reference attribute MUST be used as defined in [ESS]. The content reference MUST be a signed attribute.

The syntax of the content reference attribute type of the ES is as defined in [ESS].

3.11.2 Content Identifier Attribute

The content identifier attribute provides an identifier for the signed content for use when reference may be later required to that content, for example in the content reference attribute in other signed data sent later.

The content identifier must be a signed attribute.

The syntax of the content identifier attribute type of the ES is as defined in [ESS].

The minimal signedContentIdentifier should contain a concatenation of user-specific identification information (such as a user name or public keying material identification information), a GeneralizedTime string, and a random number.

3.11.3 Content Hints Attribute

The content hints attribute provides information that describes the format of the signed content. It may be used by the signer to indicate to a verifier the precise format that MUST be used to

present the data (e.g., text, voice, video) to a verifier. This attribute MUST be present when it is mandatory to present the signed data to human users on verification.

The syntax of the content hints attribute type of the ES is as defined in ESS (RFC 2634, section 2.9 [9]).

When used to indicate the precise format of the data to be presented to the user the following rules apply:

The contentType (defined in RFC 2630 [8]) indicates the type of the associated content. It is an object identifier (i.e., a unique string of integers) assigned by an authority that defines the content type.

The UTF8String shall define the presentation format. The format may be defined by MIME types as indicated below.

Note 1: The contentType can be id-data defined in CMS (RFC 2630 [8]). The UTF8String can be used to indicate the encoding of the data, like MIME type. RFC 2045 [25] provides a common structure for encoding a range of electronic documents and other multi-media types, see annex B for further information, a system supporting verification of electronic signature may present information to users in the form identified by the MIME type.

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

3.12 Additional Optional Attributes

3.12.1 Commitment Type Indication Attribute

There may be situation were a signer wants to explicitly indicate to a verifier that by signing the data, it illustrates a type of commitment on behalf of the signer. The commitmentTypeIndication attribute conveys such information.

The commitmentTypeIndication attribute must be a signed attribute.

The commitment type may be:

* defined as part of the signature policy, in which case the commitment type has precise semantics that is defined as part of the signature policy.

* be a registered type, in which case the commitment type has precise semantics defined by registration, under the rules of the registration authority. Such a registration authority may be a trading association or a legislative authority.

The signature policy specifies a set of attributes that it "recognizes". This "recognized" set includes all those commitment types defined as part of the signature policy as well as any externally defined commitment types that the policy may choose to recognize. Only recognized commitment types are allowed in this field.

The following object identifier identifies the commitment type indication 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}

Commitment-Type-Indication attribute values have ASN.1 type
CommitmentTypeIndication.

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

CommitmentTypeIdentifier ::= OBJECT IDENTIFIER

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

The use of any qualifiers to the commitment type is outside the scope of this document.

The following generic commitment types are defined in this document:

```
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}
```

These generic commitment types have the following meaning:

Proof of origin indicates that the signer recognizes to have created, approved and sent the message.

Proof of receipt indicates that signer recognizes to have received the content of the message.

Proof of delivery indicates that the TSP providing that indication has delivered a message in a local store accessible to the recipient of the message.

Proof of sender indicates that the entity providing that indication has sent the message (but not necessarily created it).

Proof of approval indicates that the signer has approved the content of the message.

Proof of creation indicates that the signer has created the message (but not necessarily approved, nor sent it).

3.12.2 Signer Location attribute

The signer-location attribute is an attribute which specifies a mnemonic for an address associated with the signer at a particular geographical (e.g., city) location. The mnemonic is registered in the country in which the signer is located and is used in the provision of the Public Telegram Service (according to ITU-T Recommendation F.1 [PTS]).

The signer-location attribute must be a signed attribute.

The following object identifier identifies the signer-location 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}
```

Signer-location attribute values have ASN.1 type SignerLocation.

```
SignerLocation ::= SEQUENCE {
     -- at least one of the following must 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
```

3.12.3 Signer Attributes attribute

The signer-attributes attribute is an attribute which specifies additional attributes of the signer (e.g., role).

It may be either:

* claimed attributes of the signer; or
* certified attributes of the signer;

The signer-attributes attribute must be a signed attribute.

The following object identifier identifies the signer-attribute 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}
```

signer-attribute attribute values have ASN.1 type SignerAttribute.

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

ClaimedAttributes ::= SEQUENCE OF Attribute

CertifiedAttributes ::= AttributeCertificate
         -- as defined in X.509 : see section 10.3
```

NOTE: The claimed and certified attribute are imported from ITU-T Recommendations X.501 [16] and ITU-T Recommendation X.509:Draft Amendment on Certificate Extensions, October 1999.

3.12.4 Content Time-Stamp attribute

The content time-stamp attribute is an attribute which is the time-stamp of the signed data content before it is signed.

The content time-stamp attribute must be a signed attribute.

The following object identifier identifies the signer-attribute 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}
```

Content time-stamp attribute values have ASN.1 type ContentTimestamp:
ContentTimestamp::= TimeStampToken

The value of messageImprint field within TimeStampToken must be a hash of the value of eContent field within encapContentInfo within the signedData.

For further information and definition of TimeStampToken see [TSP].

3.13 Support for Multiple Signatures

3.13.1 Independent Signatures

Multiple independent signatures are supported by independent SignerInfo from each signer.

Each SignerInfo must include all the attributes required under this document and must be processed independently by the verifier.

3.13.2 Embedded Signatures

Multiple embedded signatures are supported using the counter-signature unsigned attribute (see clause 3.10.1). Each counter signature is carried in Countersignature held as an unsigned attribute to the SignerInfo to which the counter-signature is applied.

4. Validation Data

This clause specifies the validation data structures which builds on the electronic signature specified in clause 3. This includes:

* Time-Stamp applied to the electronic signature value.

* Complete validation data which comprises the time-stamp of the signature value, plus references to all the certificates and revocation information used for full validation of the electronic signature.

The following optional eXtended forms of validation data are also defined:

* X-timestamp: There are two types of time-stamp used in extended validation data defined by this document.

  - Type 1 -Time-Stamp which comprises a time-stamp over the ES with Complete validation data (ES-C).

  - Type 2 X-Time-Stamp which comprises of a time-stamp over the certification path references and the revocation information references used to support the ES-C.

    * X-Long: This comprises a Complete validation data plus the actual values of all the certificates and revocation information used in the ES-C.

    * X-Long-Time-Stamp: This comprises a Type 1 or Type 2 X-Timestamp plus the actual values of all the certificates and revocation information used in the ES-C.

This clause also specifies the data structures used in Archive validation data:

* Archive validation data comprises a Complete validation data, the certificate and revocation values (as in a X-Long validation data), any other existing X-timestamps, plus the Signed User data and an additional archive time-stamp over all that data. An archive time-stamp may be repeatedly applied after long periods to maintain validity when electronic signature and timestamping algorithms weaken.

The additional data required to create the forms of electronic signature identified above is carried as unsigned attributes associated with an individual signature by being placed in the

unsignedAttrs field of SignerInfo. Thus all the attributes defined in clause 4 are unsigned attributes.

NOTE: Where multiple signatures are to be supported, as described in clause 3.13, each signature has a separate SignerInfo. Thus, each signature requires its own unsigned attribute values to create ES-T, ES-C etc.

4.1 Electronic Signature Timestamp

An Electronic Signature with Timestamp is an Electronic Signature for which part, but not all, of the additional data required for validation is available (e.g., some certificates and revocation information is available but not all).

The minimum structure Timestamp validation data is the Signature Timestamp Attribute as defined in clause 4.1.1 over the ES signature value.

4.1.1 Signature Timestamp Attribute Definition

The Signature Timestamp attribute is timestamp of the signature value. It is an unsigned attribute. Several instances of this attribute from different TSAs may occur with an electronic signature.

The Signature Validation Policy specifies, in the signatureTimestampDelay field of TimestampTrustConditions, a maximum acceptable time difference which is allowed between the time indicated in the signing time attribute and the time indicated by the Signature Timestamp attribute. If this delay is exceeded then the electronic signature must be considered as invalid.

The following object identifier identifies the Signature Timestamp attribute:

```
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 messageImprint field within TimeStampToken must be a hash of the value of signature field within SignerInfo for the signedData being timestamped.

For further information and definition of TimeStampToken see [TSP].

4.2 Complete Validation Data

An electronic signature with complete validation data is an Electronic Signature for which all the additional data required for validation (i.e., all certificates and revocation information) is available. Complete validation data (ES-C) build on the electronic signature Time-Stamp as defined above.

The minimum structure of a Complete validation data is:

* the Signature Time-Stamp Attribute, as defined in clause 4.1.1;
* Complete Certificate Refs, as defined in clause 4.2.1;
* Complete Revocation Refs, as defined in clause 4.2.2.

The Complete validation data MAY also include the following additional information, forming a X-Long validation data, for use if later validation processes may not have access to this information:

* Complete Certificate Values, as defined in clause 4.2.3;
* Complete Revocation Values, as defined in clause 4.2.4.

The Complete validation data MAY also include one of the following additional attributes, forming a X-Time-Stamp validation data, to provide additional protection against later CA compromise and provide integrity of the validation data used:

* ES-C Time-Stamp, as defined in clause 4.2.5; or
* Time-Stamped Certificates and CRLs references, as defined in clause 4.2.6.

NOTE 1: As long as the CA's are trusted such that these keys cannot be compromised or the cryptography used broken, the ES-C provides long term proof of a valid electronic signature.

A valid electronic signature is an electronic signature which passes validation according to a signature validation policy.

NOTE 2: The ES-C provides the following important property for long standing signatures; that is having been found once to be valid, must continue to be so months or years later. Long after the validity period of the certificates have expired, or after the user key has been compromised.

4.2.1 Complete Certificate Refs Attribute Definition

The Complete Certificate Refs attribute is an unsigned attribute. It references the full set of CA certificates that have been used to validate a ES with Complete validation data (ES-C) up to (but not including) the signer's certificate. Only a single instance of this attribute must occur with an electronic signature.

Note: The signer's certified is referenced in the signing certificate attribute (see clause 3.1).

```
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
```

OTHERCertID is defined in clause 3.8.2.

The IssuerSerial that must be present in OTHERCertID. The certHash must match the hash of the certificate referenced.

NOTE: Copies of the certificate values may be held using the Certificate Values attribute defined in clause 4.3.1.

4.2.2 Complete Revocation Refs Attribute Definition

The Complete Revocation Refs attribute is an unsigned attribute. Only a single instance of this attribute must occur with an electronic signature. It references the full set of the CRL or OCSP responses that have been used in the validation of the signer and CA certificates used in ES with Complete validation data.

The following object identifier identifies the CompleteRevocationRefs attribute:

```
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 must contain one CrlOcspRef for the signing certificate, followed by one for each OTHERCertID in the CompleteCertificateRefs attribute. The second and subsequent CrlOcspRef fields must 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.

```
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 an 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 which must correspond to 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 must be included.

The OcspIdentifier is to identify the OSCP response using the issuer name and the time of issue of the OCSP response which must correspond to the time "producedAt" contained in the issued OCSP response. Since it may be needed to make the difference between two OCSP responses received within the same second, then the hash of the response contained in the OcspResponsesID may be needed to solve the ambiguity.

NOTE: Copies of the CRL and OCSP responses values may be held using the Revocation Values attribute defined in clause 4.3.2.

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

OtherRevRefType ::= OBJECT IDENTIFIER
```

The syntax and semantics of other revocation references is outside the scope of this document. The definition of the syntax of the other form of revocation information is as identified by OtherRevRefType.

4.3 Extended Validation Data

4.3.1 Certificate Values Attribute Definition

The Certificate Values attribute is an unsigned attribute. Only a single instance of this attribute must occur with an electronic signature. It holds the values of certificates referenced in the CompleteCertificateRefs attribute.

Note: If an Attribute Certificate is used, it is not provided in this structure but must be provided by the signer as a signer-attributes attribute (see clause 12.3).

The following object identifier identifies the CertificateValues attribute:

```
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 RFC2459 and ITU-T Recommendation X.509 [1])

4.3.2 Revocation Values Attribute Definition

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

The following object identifier identifies the Revocation Values attribute:

```
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
}

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

OtherRevValType ::= OBJECT IDENTIFIER
```

The syntax and semantics of the other revocation values is outside the scope of this document. The definition of the syntax of the other form of revocation information is as identified by OtherRevRefType.

CertificateList is defined in RFC 2459 [RFC2459] and in ITU-T Recommendation X.509 [X509]).

BasicOCSPResponse is defined in RFC 2560 [OCSP].

4.3.3 ES-C Time-Stamp Attribute Definition

This attribute is used for the Type 1 X-Time-Stamped validation data. The ES-C Time-Stamp attribute is an unsigned attribute. It is time-stamp of a hash of the electronic signature and the complete validation data (ES-C). It is a special purpose TimeStampToken Attribute which time-stamps the ES-C. Several instances instance of this attribute may occur with an electronic signature from different TSAs.

The following object identifier identifies the ES-C Time-Stamp attribute:

```
id-aa-ets-escTimeStamp OBJECT IDENTIFIER ::= { iso(1) member-
body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
id-aa(2) 25}
```

The ES-C time-stamp attribute value has the ASN.1 syntax ESCTimeStampToken.

```
ESCTimeStampToken ::= TimeStampToken
```

The value of messageImprint field within TimeStampToken must be a hash of the concatenated values (without the type or length encoding for that value) of the following data objects as present in the ES with Complete validation data (ES-C):

* signature field within SignerInfo;

* SignatureTimeStampToken attribute;

* CompleteCertificateRefs attribute;

* CompleteRevocationRefs attribute.

For further information and definition of the Time Stamp Token see [TSP].

4.3.4 Time-Stamped Certificates and CRLs Attribute Definition

This attribute is used for the Type 2 X-Time-Stamp validation data. A TimestampedCertsCRLsRef attribute is an unsigned attribute. It is a list of referenced certificates and OCSP responses/CRLs which are been time-stamped to protect against certain CA compromises. Its syntax is as follows:

The following object identifier identifies the TimestampedCertsCRLsRef attribute:

```
id-aa-ets-certCRLTimestamp OBJECT IDENTIFIER ::= { iso(1) member-
body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
id-aa(2) 26}
```

The attribute value has the ASN.1 syntax TimestampedCertsCRLs.

```
TimestampedCertsCRLs ::= TimeStampToken
```

The value of messageImprint field within TimeStampToken must be a hash of the concatenated values (without the type or length encoding for that value) of the following data objects as present in the ES with Complete validation data (ES-C):

* CompleteCertificateRefs attribute;
* CompleteRevocationRefs attribute.

4.4 Archive Validation Data

Where an electronic signature is required to last for a very long time, and a the time-stamp on an electronic signature is in danger of being invalidated due to algorithm weakness or limits in the validity period of the TSA certificate, then it may be required to time-stamp the electronic signature several times. When this is required an archive time-stamp attribute may be required. This time-stamp may be repeatedly applied over a period of time.

4.4.1 Archive Time-Stamp Attribute Definition

The Archive Time-Stamp attribute is time-stamp of the user data and the entire electronic signature. If the Certificate values and Revocation Values attributes are not present these attributes must be added to the electronic signature prior to the time-stamp. The Archive Time-Stamp attribute is an unsigned attribute. Several instances of this attribute may occur with on electronic signature both over time and from different TSAs.

The following object identifier identifies the Nested Archive Time-Stamp attribute:

```
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 messageImprint field within Time-StampToken must be a hash of the concatenated values (without the type or length encoding for that value) of the following data objects as present in the electronic signature:

* encapContentInfo eContent OCTET STRING;
* signedAttributes;
* signature field within SignerInfo;
* SignatureTimeStampToken attribute;
* CompleteCertificateRefs attribute;
* CompleteRevocationData attribute;
* CertificateValues attribute
  (If not already present this information must be included in the ES-A);
* RevocationValues attribute
  (If not already present this information must be included in the ES-A);
* ESCTimeStampToken attribute if present;
* TimestampedCertsCRLs attribute if present;
* any previous ArchiveTimeStampToken attributes.

For further information and definition of TimeStampToken see [TSP]

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

5. Security Considerations

5.1 Protection of Private Key

The security of the electronic signature mechanism defined in this document depends on the privacy of the signer's private key. Implementations must take steps to ensure that private keys cannot be compromised.

5.2 Choice of Algorithms

Implementers should be aware that cryptographic algorithms become weaker with time. As new cryptoanalysis techniques are developed and computing performance improves, the work factor to break a particular cryptographic algorithm will reduce. Therefore, cryptographic algorithm implementations should be modular allowing new algorithms to be readily inserted. That is, implementers should be prepared for the set of mandatory to implement algorithms to change over time.

6. Conformance Requirements

This document only defines conformance requirements up to a ES with Complete validation data (ES-C). This means that none of the extended and archive forms of Electronic Signature (ES-X, ES-A) need to be implemented to get conformance to this standard.

This document mandates support for elements of the signature policy.

6.1 Signer

A system supporting signers according to this document must, at a minimum, support generation of an electronic signature consisting of the following components:

* The general CMS syntax and content type as defined in RFC 2630 (see clauses 4.1 and 4.2).

* CMS SignedData as defined in RFC 2630 with version set to 3 and at least one SignerInfo must be present (see clauses 4.3, 4.4, 4.5, 4.6).

* The following CMS Attributes as defined in RFC 2630:

  - ContentType; This must always be present (see clause 3.7.1);

  - MessageDigest; This must always be present (see clause 3.7.2);

  - SigningTime; This must always be present (see clause 3.7.3).

* The following ESS Attributes as defined in RFC 2634:

  - SigningCertificate: This must be set as defined in clauses 3.8.1 and 3.8.2.

* The following Attributes as defined in clause 3.9:

  - SignaturePolicyIdentifier; This must always be present.

* Public Key Certificates as defined in ITU-T Recommendation X.509 [1] and profiled in RFC 2459 [7] (see clause 9.1).

6.2 Verifier using time-stamping

A system supporting verifiers according to this document with time-stamping facilities must, at a minimum, support:

* Verification of the mandated components of an electronic signature, as defined in clause 5.1.

* Signature Time-Stamp attribute, as defined in clause 4.1.1.

* Complete Certificate Refs attribute, as defined in clause 4.2.1.

* Complete Revocation Refs Attribute, as defined in clause 4.2.2.

* Public Key Certificates, as defined in ITU-T Recommendation X.509 and profiled in RFC 2459.

* Either of:

  - Certificate Revocation Lists, as defined in ITU-T Recommendation X.509 [1] and profiled in RFC 2459 [7]; or

  - On-line Certificate Status Protocol responses, as defined in RFC 2560.

6.3 Verifier using secure records

A system supporting verifiers according to the present document shall, at a minimum, support:

* Verification of the mandated components of an electronic signature, as defined in subclause 5.1.

* Complete Certificate Refs attribute, as defined in subclause 4.2.1.

* Complete Revocation Refs Attribute, as defined in subclause 9.2.2.

* A record shall be maintained, which cannot be undetectably modified, of the electronic signature and the time when the signature was first validated using the referenced certificates and revocation information.

* Public Key Certificates, as defined in ITU-T Recommendation X.509 [1] and profiled in RFC 2459 [7] (see subclause 10.1).

* Either of:

  - Certificate Revocation Lists, as defined in ITU-T Recommendation X.509 [1] and profiled in RFC 2459 [7] Or

  - On-line Certificate Status Protocol, as defined in RFC 2560 [8] (see subclause 10.3).

7. References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[ESS] Hoffman, P., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.

[CMS] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[OCSP] Myers, M., Ankney, R., Malpani, A., Galperin, S. and C. Adams, "On-line Status Certificate Protocol", RFC 2560, June 1999.

[TSP] Adams, C., Cain, P., Pinkas, D. and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, August 2001.

[PTS] Public Telegram Service. ITU-T Recommendation F1.

[RFC2459] Housley, R., Ford, W., Polk, W. and D. Solo, "Internet X.509 Public Key Infrastructure, Certificate and CRL Profile", RFC 2459, January 1999.

[PKCS9] RSA Laboratories, "The Public-Key Cryptography Standards (PKCS)", RSA Data Security Inc., Redwood City, California, November 1993 Release.

[ISONR] ISO/IEC 10181-5: Security Frameworks in Open Systems. Non-Repudiation Framework. April 1997.

[TS101733] ETSI Standard TS 101 733 V.1.2.2 (2000-12) Electronic Signature Formats. Note: copies of ETSI TS 101 733 can be freely downloaded from the ETSI web site www.etsi.org.

8. Authors' Addresses

This Informational RFC has been produced in ETSI TC-SEC.

ETSI
F-06921 Sophia Antipolis, Cedex - FRANCE
650 Route des Lucioles - Sophia Antipolis
Valbonne - France
Tel: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16
secretariat@etsi.fr
http://www.etsi.org

Contact Point

Harri Rasilainen
ETSI
650 Route des Lucioles
F-06921 Sophia Antipolis, Cedex
FRANCE

EMail: harri.rasilainen@etsi.fr

Denis Pinkas
Integris
68, Route de Versailles
78434 Louveciennes CEDEX
FRANCE

EMail: Denis.Pinkas@bull.net

John Ross
Security & Standards
192 Moulsham Street
Chelmsford, Essex
CM2 0LG
United Kingdom

EMail: ross@secstan.com

Nick Pope
Security & Standards
192 Moulsham Street
Chelmsford, Essex
CM2 0LG
United Kingdom

EMail: pope@secstan.com

Annex A (normative): ASN.1 Definitions

This annex provides a summary of all the ASN.1 syntax definitions for new syntax defined in this document.

A.1 Definitions Using X.208 (1988) ASN.1 Syntax

NOTE: The ASN.1 module defined in clause A.1 has precedence over that defined in Annex A-2 in the case of any conflict.

```
   ETS-ElectronicSignatureFormats-88syntax { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-mod(0) 5}

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS All -

IMPORTS

-- Crypographic Message Syntax (CMS): RFC 2630

  ContentInfo, ContentType, id-data, id-signedData, SignedData,
  EncapsulatedContentInfo, SignerInfo, id-contentType,
  id-messageDigest, MessageDigest, id-signingTime, SigningTime,
  id-countersignature, Countersignature

  FROM CryptographicMessageSyntax
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
      smime(16) modules(0) cms(1) }

-- ESS Defined attributes: RFC 2634
-- (Enhanced Security Services for S/MIME)

  id-aa-signingCertificate, SigningCertificate, IssuerSerial,
  id-aa-contentReference, ContentReference,
  id-aa-contentIdentifier, ContentIdentifier

  FROM ExtendedSecurityServices
     { iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs-9(9) smime(16) modules(0) ess(2) }

-- Internet X.509 Public Key Infrastructure
-- Certificate and CRL Profile: RFC 2459

  Certificate, AlgorithmIdentifier, CertificateList, Name,
  GeneralNames, GeneralName, DirectoryString,Attribute,
```

```
  AttributeTypeAndValue, AttributeType, AttributeValue,
  PolicyInformation, BMPString, UTF8String

  FROM PKIX1Explicit88
  {iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit-
   88(1)}

-- X.509 '97 Authentication Framework

AttributeCertificate

  FROM AuthenticationFramework
  {joint-iso-ccitt ds(5) module(1) authenticationFramework(7) 3}

-- The imported AttributeCertificate is defined using the X.680 1997
-- ASN.1 Syntax,
-- an equivalent using the 88 ASN.1 syntax may be used.


-- OCSP 2560

BasicOCSPResponse, ResponderID

  FROM OCSP {-- OID not assigned -- }

-- Time Stamp Protocol Work in Progress

TimeStampToken

  FROM PKIXTSP
  {iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-tsp(13)}

-- S/MIME Object Identifier arcs used in this document
-- ===================================================

-- S/MIME  OID arc used in this document
-- id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2)
--             us(840) rsadsi(113549) pkcs(1) pkcs-9(9) 16 }

-- S/MIME Arcs
-- id-mod  OBJECT IDENTIFIER ::= { id-smime 0 }
-- modules
-- id-ct   OBJECT IDENTIFIER ::= { id-smime 1 }
-- content types
-- id-aa   OBJECT IDENTIFIER ::= { id-smime 2 }
-- attributes
```

```
-- id-spq  OBJECT IDENTIFIER ::= { id-smime 5 }
-- signature policy qualifier
-- id-cti  OBJECT IDENTIFIER ::= { id-smime 6 }
-- commitment type identifier

-- Definitions of Object Identifier arcs used in this document
-- ===========================================================

-- The allocation of OIDs to specific objects are given below with the
-- associated ASN.1 syntax definition

-- OID used referencing electronic signature mechanisms based on this
-- standard for use with the IDUP API (see annex D)

id-etsi-es-IDUP-Mechanism-v1 OBJECT IDENTIFIER ::=
  { itu-t(0) identified-organization(4) etsi(0)
     electronic-signature-standard (1733) part1 (1)
         idupMechanism (4)etsiESv1(1) }

-- CMS Attributes Defined in this document
-- =======================================

-- Mandatory Electronic Signature Attributes

-- OtherSigningCertificate

    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 IN THIS DOCUMENT
}

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

OtherHash ::= CHOICE {
    sha1Hash     OtherHashValue,  -- This contains a SHA-1 hash
    otherHash    OtherHashAlgAndValue
}

OtherHashValue ::= OCTET STRING
```

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

-- Signature Policy Identifier

    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 }

"SignaturePolicy 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))
}

-- Optional Electronic Signature Attributes

-- Commitment Type

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}

-- Signer Location

   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 must 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

-- Signer Attributes

    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  -- as defined in X.509 :
see section 10.3

-- Content Time-Stamp

    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

-- Validation Data

-- Signature Time-Stamp

    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}

SignatureTimeStampToken ::= TimeStampToken

-- Complete Certificate Refs.

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}

CompleteCertificateRefs ::=  SEQUENCE OF OTHERCertID

-- Complete Revocation Refs

   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}

CompleteRevocationRefs ::=  SEQUENCE OF CrlOcspRef

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

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
}

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

OtherRevRefType ::= OBJECT IDENTIFIER

-- Certificate Values

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}

CertificateValues ::=  SEQUENCE OF Certificate

-- Certificate Revocation Values

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}

RevocationValues ::=  SEQUENCE {
   crlVals          [0] SEQUENCE OF CertificateList     OPTIONAL,
   ocspVals         [1] SEQUENCE OF BasicOCSPResponse   OPTIONAL,
   otherRevVals     [2] OtherRevVals
}

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

OtherRevValType ::= OBJECT IDENTIFIER

-- ES-C Time-Stamp
```

```
id-aa-ets-escTimeStamp OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 25}

ESCTimeStampToken ::= TimeStampToken

-- Time-Stamped Certificates and CRLs

id-aa-ets-certCRLTimestamp OBJECT IDENTIFIER ::= { iso(1) member-
    body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16)
    id-aa(2) 26}

TimestampedCertsCRLs ::= TimeStampToken

-- Archive Time-Stamp

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}

ArchiveTimeStampToken ::= TimeStampToken

END -- ETS-ElectronicSignatureFormats-88syntax --
```

A.2 Definitions Using X.680 1997 ASN.1 Syntax

NOTE: The ASN.1 module defined in clause A.1 has precedence over that defined in clause A.2 in the case of any conflict.

```
      ETS-ElectronicSignatureFormats-97Syntax { iso(1) member-body(2)
      us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-mod(0) 6}

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS All -

IMPORTS

-- Cryptographic Message Syntax (CMS): RFC 2630

  ContentInfo, ContentType, id-data, id-signedData, SignedData,
  EncapsulatedContentInfo, SignerInfo, id-contentType,
  id-messageDigest, MessageDigest, id-signingTime,
  SigningTime, id-countersignature, Countersignature

   FROM CryptographicMessageSyntax
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
```

```
    smime(16) modules(0) cms(1) }


-- ESS Defined attributes: RFC 2634 (Enhanced Security Services
-- for S/MIME)

   id-aa-signingCertificate, SigningCertificate, IssuerSerial,
   id-aa-contentReference, ContentReference,
   id-aa-contentIdentifier, ContentIdentifier

  FROM ExtendedSecurityServices
    { iso(1) member-body(2) us(840) rsadsi(113549)
       pkcs(1) pkcs-9(9) smime(16) modules(0) ess(2) }

-- Internet X.509 Public Key Infrastructure
- - Certificate and CRL Profile:RFC 2459

   Certificate, AlgorithmIdentifier, CertificateList, Name,
   GeneralNames, GeneralName, DirectoryString, Attribute,
   AttributeTypeAndValue, AttributeType, AttributeValue,
   PolicyInformation.


  FROM PKIX1Explicit93
    {iso(1) identified-organization(3) dod(6) internet(1)
     security(5) mechanisms(5) pkix(7) id-mod(0)
     id-pkix1-explicit-88(1)}

-- X.509 '97 Authentication Framework

        AttributeCertificate

        FROM AuthenticationFramework
        {joint-iso-ccitt ds(5) module(1) authenticationFramework(7) 3}

-- OCSP 2560

      BasicOCSPResponse, ResponderID

  FROM OCSP

--  { OID not assigned }

-- Time Stamp Protocol Work in Progress TimeStampToken

  FROM PKIXTSP
  {iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-tsp(13)}
```

```
-- S/MIME Object Identifier arcs used in this document
-- ===================================================

-- S/MIME  OID arc used in this document
-- id-smime OBJECT IDENTIFIER ::= { iso(1) member-body(2)
--             us(840) rsadsi(113549) pkcs(1) pkcs-9(9) 16 }

-- S/MIME Arcs
-- id-mod  OBJECT IDENTIFIER ::= { id-smime 0 }
-- modules
-- id-ct   OBJECT IDENTIFIER ::= { id-smime 1 }
-- content types
-- id-aa   OBJECT IDENTIFIER ::= { id-smime 2 }
-- attributes
-- id-spq  OBJECT IDENTIFIER ::= { id-smime 5 }
-- signature policy qualifier
-- id-cti  OBJECT IDENTIFIER ::= { id-smime 6 }
-- commitment type identifier

-- Definitions of Object Identifier arcs used in this document
-- ===========================================================

-- The allocation of OIDs to specific objects are given below with the
-- associated ASN.1 syntax definition

-- OID used referencing electronic signature mechanisms based on this
-- standard for use with the IDUP API (see annex D)

id-etsi-es-IDUP-Mechanism-v1 OBJECT IDENTIFIER ::=
  { itu-t(0) identified-organization(4) etsi(0)
   electronic-signature-standard (1733) part1 (1)
   idupMechanism (4)etsiESv1(1) }

-- CMS Attributes Defined in this document
-- =======================================

-- Mandatory Electronic Signature Attributes
-- OtherSigningCertificate

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 IN THIS DOCUMENT
}
```

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

OtherHash ::= CHOICE {
    sha1Hash OtherHashValue,  -- This contains a SHA-1 hash
    otherHash OtherHashAlgAndValue
}

OtherHashValue ::= OCTET STRING

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

-- Signature Policy Identifier

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 }

"SignaturePolicy 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    SIG-POLICY-QUALIFIER.&id
                                 ({SupportedSigPolicyQualifiers}),
        qualifier               SIG-POLICY-QUALIFIER.&Qualifier
                                ({SupportedSigPolicyQualifiers}
                                 {@sigPolicyQualifierId})OPTIONAL }
```

```
SupportedSigPolicyQualifiers SIG-POLICY-QUALIFIER ::=
                          { noticeToUser | pointerToSigPolSpec }

SIG-POLICY-QUALIFIER ::= CLASS {
        &id             OBJECT IDENTIFIER UNIQUE,
        &Qualifier      OPTIONAL }

WITH SYNTAX {
        SIG-POLICY-QUALIFIER-ID     &id
        [SIG-QUALIFIER-TYPE &Qualifier] }

noticeToUser SIG-POLICY-QUALIFIER ::= {
      SIG-POLICY-QUALIFIER-ID id-sqt-unotice SIG-QUALIFIER-TYPE
                                              SPUserNotice
                                                    }

pointerToSigPolSpec SIG-POLICY-QUALIFIER ::= {
      SIG-POLICY-QUALIFIER-ID id-sqt-uri SIG-QUALIFIER-TYPE SPuri }

    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))
}

-- Optional Electronic Signature Attributes

-- Commitment Type
```

```
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 {
        commitmentQualifierId       COMMITMENT-QUALIFIER.&id,
        qualifier                   COMMITMENT-QUALIFIER.&Qualifier
                                              OPTIONAL }

COMMITMENT-QUALIFIER ::= CLASS {
                    &id             OBJECT IDENTIFIER UNIQUE,
                    &Qualifier      OPTIONAL }
WITH SYNTAX {
         COMMITMENT-QUALIFIER-ID     &id
                        [COMMITMENT-TYPE &Qualifier] }

  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}

-- Signer Location
```

```
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 must 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

-- Signer Attributes

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
-- As defined in X.509 : see section 10.3

-- Content Time-Stamp

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

-- Validation Data

-- Signature Time-Stamp

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}

SignatureTimeStampToken ::= TimeStampToken

-- Complete Certificate Refs.

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}

CompleteCertificateRefs ::=  SEQUENCE OF OTHERCertID

-- Complete Revocation Refs

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}

CompleteRevocationRefs ::=  SEQUENCE OF CrlOcspRef

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

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
                                        }

OtherRevRefs ::= SEQUENCE {
   otherRevRefType  OTHER-REVOCATION-REF.&id,
  otherRevRefs  OTHER-REVOCATION-REF.&Type
```

```
                                                    }

OTHER-REVOCATION-REF ::= CLASS {
    &Type,
    &id  OBJECT IDENTIFIER UNIQUE }
  WITH SYNTAX {
    &Type ID &id }

-- Certificate Values

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}

CertificateValues ::=  SEQUENCE OF Certificate

-- Certificate Revocation Values

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}

RevocationValues ::=  SEQUENCE {
   crlVals          [0] SEQUENCE OF CertificateList OPTIONAL,
   ocspVals         [1] SEQUENCE OF BasicOCSPResponse OPTIONAL,
   otherRevVals      [2] OtherRevVals }

OtherRevVals ::= SEQUENCE {
   otherRevValType  OTHER-REVOCATION-VAL.&id,
  otherRevVals  OTHER-REVOCATION-VAL.&Type
                                                    }

OTHER-REVOCATION-VAL ::= CLASS {
    &Type,
    &id  OBJECT IDENTIFIER UNIQUE }
  WITH SYNTAX {
    &Type ID &id }

-- ES-C Time-Stamp

id-aa-ets-escTimeStamp OBJECT IDENTIFIER ::= { iso(1)
     member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
     smime(16) id-aa(2) 25}

ESCTimeStampToken ::= TimeStampToken

-- Time-Stamped Certificates and CRLs

id-aa-ets-certCRLTimestamp OBJECT IDENTIFIER ::= { iso(1)
```

```
    member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
    smime(16) id-aa(2) 26}

TimestampedCertsCRLs ::= TimeStampToken

-- Archive Time-Stamp

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}

ArchiveTimeStampToken ::= TimeStampToken

END                -- ETS-ElectronicSignatureFormats-97Syntax
```

Annex B (informative): General Description

This annex captures the concepts that apply to this document and the rational for the elements of the specification defined using ASN.1 in the main text of this document.

The specification below includes a description why the component is needed, with a brief description of the vulnerabilities and threats and the manner by which they are countered.

B.1 The Signature Policy

The signature policy is a set of rules for the creation and validation of an electronic signature, under which the signature can be determined to be valid. A given legal/contractual context may recognize a particular signature policy as meeting its requirements. A signature policy may be issued, for example, by a party relying on the electronic signatures and selected by the signer for use with that relying party. Alternatively, a signature policy may be established through an electronic trading association for use amongst its members. Both the signer and verifier use the same signature policy.

The signature policy may be explicitly identified or may be implied by the semantics of the data being signed and other external data like a contract being referenced which itself refers to a signature policy.

An explicit signature policy has a globally unique reference, which is bound to an electronic signature by the signer as part of the signature calculation.

The signature policy needs to be available in human readable form so that it can be assessed to meet the requirements of the legal and contractual context in which it is being applied. To facilitate the automatic processing of an electronic signature the parts of the signature policy which specify the electronic rules for the creation and validation of the electronic signature also needs to be in a computer processable form.

The signature policy thus includes the following:

* Information about the signature policy that can be displayed to the signer or the verifiers.
* Rules, which apply to functionality, covered by this document (referred to as the Signature Validation Policy).
* Rules which may be implied through adoption of Certificate Policies that apply to the electronic signature (e.g., rules for ensuring the secrecy of the private signing key).
* Rules, which relate to the environment used by the signer, e.g., the use of an agreed CAD (Card Accepting Device) used in conjunction with a smart card.

An explicit Signature Validation Policy may be structured so that it can be computer processable. Any format of the signature validation policy is allowed by this document. However, for a given explicit signature policy there must be one definitive form that has a unique binary encoded value.

The Signature Validation Policy includes rules regarding use of TSPs (CA, Attribute Authorities, Time Stamping Authorities) as well as rules defining the components of the electronic signature that must be provided by the signer with data required by the verifier to provide long term proof.

B.2 Signed Information

The information being signed may be defined as a MIME-encapsulated message which can be used to signal the format of the content in order to select the right display or application. It can be composed of formatted text (e.g., EDIFACT), free text or of fields from an electronic form (e-form). For example, the Adobe(tm) format "pdf" may be used or the eXtensible Mark up Language (XML).

B.3 Components of an Electronic Signature

B.3.1 Reference to the Signature Policy

The definition of electronic signature includes: "a commitment has been explicitly endorsed under a "Signature policy", at a given time, by a signer under an identifier, e.g., a name or a pseudonym, and optionally a role".

When two independent parties want to evaluate an electronic signature, it is fundamental that they get the same result. To meet this requirement same signature policy must be used by the signer and verifier.

The signature policy may be explicitly identified or may be implied by the semantics of the data being signed and other external data which designate the signature policy to be used.

By signing over the signature policy identifier the signer explicitly indicates that he or she has applied the signature policy in creating the signature. Thus, undertakes any explicit or implied commitments.

In order to unambiguously identify an explicit signature policy that is to be used to verify the signature an identifier and hash of the "Signature policy" shall be part of the signed data. Additional information about the explicit policy (e.g., web reference to the document) may be carried as "qualifiers" to the signature policy identifier.

When the signature policy not explicitly identified, but is implied by the semantics of the data being signed, then the signature will include a signature policy identifier that indicates that the signature policy is implied. In this case the verification rules must be determined by using other external data which will designate the signature policy to be used. If it may be determined from the context that all the documents to be verified refer to the same signature policy, then that policy may be predetermined or fixed within the application.

In order to identify unambiguously the "Signature Validation Policy" to be used to verify the signature an identifier and hash of the "Signature policy" must be part of the signed data. Additional information about the policy (e.g., web reference to the document) may be carried as "qualifiers" to the signature policy identifier.

B.3.2 Commitment Type Indication

The definition of electronic signature includes: "a commitment has been explicitly endorsed under a signature policy, at a given time, by a signer under an identifier, e.g., a name or a pseudonym, and optionally a role".

The commitment type can be indicated in the electronic signature either:

* explicitly using a "commitment type indication" in the electronic signature;

* implicitly or explicitly from the semantics of the signed data.

If the indicated commitment type is explicit using a "commitment type indication" in the electronic signature, acceptance of a verified signature implies acceptance of the semantics of that commitment type. The semantics of explicit commitment types indications must be specified either as part of the signature policy or may be registered for generic use across multiple policies.

If a signature includes a commitment type indication other than one of those recognized under the signature policy the signature must be treated as invalid.

How commitment is indicated using the semantics of the data being signed is outside the scope of this document.

NOTE: Examples of commitment indicated through the semantics of the data being signed, are:

* An explicit commitment made by the signer indicated by the type of data being signed over. Thus, the data structure being signed can have an explicit commitment within the context of the application (e.g., EDIFACT purchase order).

* An implicit commitment which is a commitment made by the signer because the data being signed over has specific semantics (meaning) which is only interpretable by humans, (i.e., free text).

B.3.3 Certificate Identifier from the Signer

The definition of the ETSI electronic signature includes: "a commitment has been explicitly endorsed under a signature policy, at a given time, by a signer under an identifier, e.g., a name or a pseudonym, and optionally a role."

In many real life environments users will be able to get from different CAs or even from the same CA, different certificates containing the same public key for different names. The prime advantage is that a user can use the same private key for different purposes. Multiple use of the private key is an advantage when a smart card is used to protect the private key, since the storage of a smart card is always limited. When several CAs are involved, each different certificate may contain a different identity, e.g., as a national or as an employee from a company. Thus when a private key is used for various purposes, the certificate is needed to clarify the context in which the private key was used when generating the signature. Where there is the possibility of multiple use of private keys it is necessary for the signer to indicate to the verifier the precise certificate to be used.

Many current schemes simply add the certificate after the signed data and thus are subject to various substitution attacks. An example of a substitution attack is a "bad" CA that would issue a certificate to someone with the public key of someone else. If the certificate from the signer was simply appended to the signature and thus not protected by the signature, any one could substitute one certificate by another and the message would appear to be signed by some one else.

In order to counter this kind of attack, the identifier of the signer has to be protected by the digital signature from the signer.

Although it does not provide the same advantages as the previous technique, another technique to counter that threat has been identified. It requires all CAs to perform a Proof Of Possession of the private key at the time of registration. The problem with that technique is that it does not provide any guarantee at the time of verification and only some proof "after the event" may be obtained, if and only if the CA keeps the Proof Of Possession in audit trail.

In order to identify unambiguously the certificate to be used for the verification of the signature an identifier of the certificate from the signer must be part of the signed data.

B.3.4 Role Attributes

The definition of electronic signature includes: "a commitment has been explicitly endorsed under a non repudiation security policy, at a given time, by a signer under an identifier, e.g., a name or a pseudonym, and optionally a role."

While the name of the signer is important, the position of the signer within a company or an organization can be even more important. Some contracts may only be valid if signed by a user in a particular role, e.g., a Sales Director. In many cases whom the sales Director really is, is not that important but being sure that the signer is empowered by his company to be the Sales Director is fundamental.

This document defines two different ways for providing this feature:

* by placing a claimed role name in the CMS signed attributes field;

* by placing a attribute certificate containing a certified role name in the CMS signed attributes field.

NOTE: Another possible approach would have been to use additional attributes containing the roles name(s) in the signer's certificate. However, it was decided not to follow this approach as it breaks the basic philosophy of the certificate being issued for one primary purpose. Also, by using separate certificates for management of the signer's identity certificate and management of additional roles can simplify the management, as new identity keys need not be issued if a use of role is to be changed.

B.3.4.1 Claimed Role

The signer may be trusted to state his own role without any certificate to corroborate this claim. In which case the claimed role can be added to the signature as a signed attribute.

B.3.4.2 Certified Role

Unlike public key certificates that bind an identifier to a public key, Attribute Certificates bind the identifier of a certificate to some attributes, like a role. An Attribute Certificate is NOT issued by a CA but by an Attribute Authority (AA). The Attribute Authority will be most of the time under the control of an organization or a company that is best placed to know which attributes are relevant for which individual.

The Attribute Authority may use or point to public key certificates issued by any CA, provided that the appropriate trust may be placed in that CA. Attribute Certificates may have various periods of validity. That period may be quite short, e.g., one day. While this requires that a new Attribute Certificate is obtained every day, valid for that day, this can be advantageous since revocation of such certificates may not be needed. When signing, the signer will have to specify which Attribute Certificate it selects. In order to do

so, a reference to the Attribute Certificate will have to be included in the signed data in order to be protected by the digital signature from the signer.

In order to identify unambiguously the attribute certificate(s) to be used for the verification of the signature an identifier of the attribute certificate(s) from the signer must be part of the signed data.

B.3.5 Signer Location

In some transactions the purported location of the signer at the time he or she applies his signature may need to be indicated. For this reason an optional location indicator must be able to be included.

In order to provide indication of the location of the signer at the time he or she applied his signature a location attribute may be included in the signature.

B.3.6 Signing Time

The definition of electronic signature includes: "a commitment has been explicitly endorsed under a signature policy, at a given time, by a signer under an identifier, e.g., a name or a pseudonym, and optionally a role."

There are several ways to address this problem. The solution adopted in this document is to sign over a time which the signer claims is the signing time (i.e., claimed signing time) and to require a trusted time stamp to be obtained when building a ES with Time-Stamp. When a verifier accepts a signature, the two times must be within acceptable limits.

The solution that is adopted in this document offers the major advantage that electronic signatures can be generated without any on-line connection to a trusted time source (i.e., they may be generated off-line).

Thus two dates and two signatures are required:

* a signing time indicated by the signer and which is part of the data signed by the signer (i.e., part of the basic electronic signature);

* a time indicated by a Time-Stamping Authority (TSA) which is signed over the digital signature value of the basic electronic signature. The signer, verifier or both may obtain the TSA time-stamp.

In order for an electronic signature to be valid under a signature policy, it must be time-stamped by a TSA where the signing time as indicated by the signer and the time of time stamping as indicated by a TSA must be "close enough" to meet the requirements of the signature validation policy.

"Close enough" means a few minutes, hours or even days according to the "Signature Validation Policy".

NOTE: The need for Time-Stamping is further explained in clause B.4.5. A further optional attribute is defined in this document to time-stamp the content, to provide proof of the existence of the content, at the time indicated by the time-stamp.

Using this optional attribute a trusted secure time may be obtained before the document is signed and included under the digital signature. This solution requires an on-line connection to a trusted time-stamping service before generating the signature and may not represent the precise signing time, since it can be obtained in advance. However, this optional attribute may be used by the signer to prove that the signed object existed before the date included in the time-stamp (see 3.12.3, Content Time-Stamp).

Also, the signing time should be between the time indicated by this time-stamp and time indicated by the ES-T time-stamp.

B.3.7 Content Format

When presenting signed data to a human user it may be important that there is no ambiguity as to the presentation of the signed information to the relying party. In order for the appropriate representation (text, sound or video) to be selected by the relying party a content hint may be indicated by the signer. If a relying party system does not use the format specified in the content hints to present the data to the relying party, the electronic signature may not be valid.

B.4 Components of Validation Data

B.4.1 Revocation Status Information

A verifier will have to prove that the certificate of the signer was valid at the time of the signature. This can be done by either:

* using Certificate Revocation Lists (CRLs);

* using responses from an on-line certificate status server (for example; obtained through the OCSP protocol).

B.4.2 CRL Information

When using CRLs to get revocation information, a verifier will have to make sure that he or she gets at the time of the first verification the appropriate certificate revocation information from the signer's CA. This should be done as soon as possible to minimize the time delay between the generation and verification of the signature. This involves checking that the signer certificate serial number is not included in the CRL. The signer, the verifier or any other third party may obtain either this CRL. If obtained by the signer, then it must be conveyed to the verifier. It may be convenient to archive the CRL for ease of subsequent verification or arbitration.

Alternatively, provided the CRL is archived elsewhere which is accessible for the purpose of arbitration, then the serial number of the CRL used may be archived together with the verified electronic signature.

It may happen that the certificate serial number appears in the CRL but with the status "suspended" (i.e., on hold). In such a case, the electronic signature is not yet valid, since it is not possible to know whether the certificate will or will not be revoked at the end of the suspension period. If a decision has to be taken immediately then the signature has to be considered as invalid. If a decision can wait until the end of the suspension period, then two cases are possible:

* the certificate serial number has disappeared from the list and thus the certificate can be considered as valid and that CRL must be captured and archived either by the verifier or elsewhere and be kept accessible for the purpose of arbitration.

* the certificate serial number has been maintained on the list with the status definitively revoked and thus the electronic signature must be considered as invalid and discarded.

At this point the verifier may be convinced that he or she got a valid signature, but is not yet in a position to prove at a later time that the signature was verified as valid. Before addressing this point, an alternative to CRL is to use OCSP responses.

B.4.3 OCSP Information

When using OCSP to get revocation information , a verifier will have to make sure that he or she gets at the time of the first verification an OCSP response that contains the status "valid". This

should be done as soon as possible after the generation of the signature. The signer, the verifier or any other third party may fetch this OCSP response. Since OSCP responses are transient and thus are not archived by any TSP including CA, it is the responsibility of every verifier to make sure that it is stored in a safe place. The simplest way is to store them associated with the electronic signature. An alternative would be to store them in some storage so that they can then be easily retrieved.

In the same way as for the case of the CRL, it may happen that the certificate is declared as invalid but with the secondary status "suspended".

In such a case, the electronic signature is not yet valid, since it is not possible to know whether the certificate will or will not be revoked at the end of the suspension period. If a decision has to be taken immediately then the electronic signature has to be considered as invalid. If a decision can wait until the end of the suspension period, then two cases are possible:

* An OCSP response with a valid status is obtained at a later date and thus the certificate can be considered as valid and that OCSP response must be captured.

* An OCSP response with an invalid status is obtained with a secondary status indicating that the certificate is definitively revoked and thus the electronic signature must be considered as invalid and discarded.

As in the CRL case, at this point, the verifier may be convinced that he or she got a valid signature, but is not yet in a position to prove at a later time that the signature was verified as valid.

B.4.4 Certification Path

A verifier will have to prove that the certification path was valid, at the time of the signature, up to a trust point according to the naming constraints and the certificate policy constraints from the "Signature Validation Policy". It will be necessary to capture all the certificates from the certification path, starting with those from the signer and ending up with those of the self-signed certificate from one trusted root of the "Signature Validation Policy". In addition, it will be necessary to capture the Authority Revocation Lists (ARLs) to prove than none of the CAs from the chain was revoked at the time of the signature.

As in the OCSP case, at this point, the verifier may be convinced that he or she got a valid signature, but is not yet in a position to prove at a later time that the signature was verified as valid.

B.4.5 Time-Stamping for Long Life of Signature

An important property for long standing signatures is that a signature, having been found once to be valid, must continue to be so months or years later.

A signer, verifier or both may be required to provide on request, proof that a digital signature was created or verified during the validity period of the all the certificates that make up the certificate path. In this case, the signer, verifier or both will also be required to provide proof that all the user and CA certificates used were not revoked when the signature was created or verified.

It would be quite unacceptable, to consider a signature as invalid even if the keys or certificates were later compromised. Thus there is a need to be able to demonstrate that the signature keys was valid around the time that the signature was created to provide long term evidence of the validity of a signature.

It could be the case that a certificate was valid at the time of the signature but revoked some time later. In this event, evidence must be provided that the document was signed before the signing key was revoked.

Time-Stamping by a Time Stamping Authority (TSA) can provide such evidence. A time stamp is obtained by sending the hash value of the given data to the TSA. The returned "time-stamp" is a signed document that contains the hash value, the identity of the TSA, and the time of stamping. This proves that the given data existed before the time of stamping. Time-Stamping a digital signature (by sending a hash of the signature to the TSA) before the revocation of the signer's private key, provides evidence that the signature has been created before the key was revoked.

If a recipient wants to hold a valid electronic signature he will have to ensure that he has obtained a valid time stamp for it, before that key (and any key involved in the validation) is revoked. The sooner the time-stamp is obtained after the signing time, the better.

It is important to note that signatures may be generated "off-line" and time-stamped at a later time by anyone, for example by the signer or any recipient interested in the value of the signature. The time stamp can thus be provided by the signer together with the signed

document, or obtained by the recipient following receipt of the signed document.

The time stamp is NOT a component of the Electronic Signature, but the essential component of the ES with Time-Stamp.

It is required in this document that signer's digital signature value is time-stamped by a trusted source, known as a Time-Stamping Authority.

This document requires that the signer's digital signature value is time-stamped by a trusted source before the electronic signature can become a ES with Complete validation data (ES-C). The acceptable TSAs are specified in the Signature Validation Policy.

Should both the signer and verifier be required to time-stamp the signature value to meet the requirements of the signature policy, the signature policy MAY specify a permitted time delay between the two time stamps.

B.4.6 Time-Stamping before CA Key Compromises

Time-Stamped extended electronic signatures are needed when there is a requirement to safeguard against the possibility of a CA key in the certificate chain ever being compromised. A verifier may be required to provide on request, proof that the certification path and the revocation information used a the time of the signature were valid, even in the case where one of the issuing keys or OCSP responder keys is later compromised.

The current document defines two ways of using time-stamps to protect against this compromise:

* Time-Stamp the ES with Complete validation data, when an OCSP response is used to get the status of the certificate from the signer.

* Time-Stamp only the certification path and revocation information references when a CRL is used to get the status of the certificate from the signer.

NOTE: the signer, verifier or both may obtain the time-stamp.

B.4.6.1 Time-Stamping the ES with Complete validation data

When an OCSP response is used, it is necessary to time stamp in particular that response in the case the key from the responder would be compromised. Since the information contained in the OCSP response

is user specific and time specific, an individual time stamp is needed for every signature received. Instead of placing the time stamp only over the certification path references and the revocation information references, which include the OCSP response, the time stamp is placed on the ES-C. Since the certification path and revocation information references are included in the ES with Complete validation data they are also protected. For the same cryptographic price, this provides an integrity mechanism over the ES with Complete validation data. Any modification can be immediately detected. It should be noticed that other means of protecting/detecting the integrity of the ES with Complete Validation Data exist and could be used.

Although the technique requires a time stamp for every signature, it is well suited for individual users wishing to have an integrity protected copy of all the validated signatures they have received.

By time-stamping the complete electronic signature, including the digital signature as well as the references to the certificates and revocation status information used to support validation of that signature, the time-stamp ensures that there is no ambiguity in the means of validating that signature.

This technique is referred to as ES with eXtended validation data (ES-X), type 1 Time-Stamped in this document.

NOTE: Trust is achieved in the references by including a hash of the data being referenced.

If it is desired for any reason to keep a copy of the additional data being referenced, the additional data may be attached to the electronic signature, in which case the electronic signature becomes a ES-X Long as defined by this document.

A ES-X Long Time-Stamped is simply the concatenation of a ES-X Time-Stamped with a copy of the additional data being referenced.

B.4.6.2 Time-Stamping Certificates and Revocation Information

References Time-Stamping each ES with Complete validation data as defined above may not be efficient, particularly when the same set of CA certificates and CRL information is used to validate many signatures.

Time-Stamping CA certificates will stop any attacker from issuing bogus CA certificates that could be claimed to existing before the CA key was compromised. Any bogus time-stamped CA certificates will show that the certificate was created after the legitimate CA key was

compromised. In the same way, time-stamping CA CRLs, will stop any attacker from issuing bogus CA CRLs which could be claimed to existing before the CA key was compromised.

Time-Stamping of commonly used certificates and CRLs can be done centrally, e.g., inside a company or by a service provider. This method reduces the amount of data the verifier has to time-stamp, for example it could reduce to just one time stamp per day (i.e., in the case were all the signers use the same CA and the CRL applies for the whole day). The information that needs to be time stamped is not the actual certificates and CRLs but the unambiguous references to those certificates and CRLs.

To comply with extended validation data, type 2 Time-stamped, this document requires the following:

* All the CA certificates references and revocation information references (i.e., CRLs) used in validating the ES-C are covered by one or more time-stamp.

Thus a ES-C with a time-stamp signature value at time T1, can be proved valid if all the CA and CRL references are time-stamped at time T1+.

B.4.7 Time-Stamping for Long Life of Signature

Advances in computing increase the probability of being able to break algorithms and compromise keys. There is therefore a requirement to be able to protect electronic signatures against this probability.

Over a period of time weaknesses may occur in the cryptographic algorithms used to create an electronic signature (e.g., due to the time available for cryptoanalysis, or improvements in cryptoanalytical techniques). Before this such weaknesses become likely, a verifier should take extra measures to maintain the validity of the electronic signature. Several techniques could be used to achieve this goal depending on the nature of the weakened cryptography. In order to simplify, a single technique, called Archive validation data, covering all the cases is being used in this document.

Archive validation data consists of the Complete validation data and the complete certificate and revocation data, time stamped together with the electronic signature. The Archive validation data is necessary if the hash function and the crypto algorithms that were used to create the signature are no longer secure. Also, if it

cannot be assumed that the hash function used by the Time Stamping Authority is secure, then nested time-stamps of Archived Electronic Signature are required.

The potential for Trusted Service Provider (TSP) key compromise should be significantly lower than user keys, because TSP(s) are expected to use stronger cryptography and better key protection. It can be expected that new algorithms (or old ones with greater key lengths) will be used. In such a case, a sequence of time-stamps will protect against forgery. Each time-stamp needs to be affixed before either the compromise of the signing key or of the cracking of the algorithms used by the TSA. TSAs (Time-Stamping Authorities) should have long keys (e.g., which at the time of drafting this document was 2048 bits for the signing RSA algorithm) and/or a "good" or different algorithm.

Nested time-stamps will also protect the verifier against key compromise or cracking the algorithm on the old electronic signatures.

The process will need to be performed and iterated before the cryptographic algorithms used for generating the previous time stamp are no longer secure. Archive validation data may thus bear multiple embedded time stamps.

B.4.8 Reference to Additional Data

Using type 1 or 2 of Time-Stamped extended validation data verifiers still needs to keep track of all the components that were used to validate the signature, in order to be able to retrieve them again later on. These components may be archived by an external source like a trusted service provider, in which case referenced information that is provided as part of the ES with Complete validation data (ES-C) is adequate. The actual certificates and CRL information reference in the ES-C can be gathered when needed for arbitration.

B.4.9 Time-Stamping for Mutual Recognition

In some business scenarios both the signer and the verifier need to time-stamp their own copy of the signature value. Ideally the two time-stamps should be as close as possible to each other.

Example: A contract is signed by two parties A and B representing their respective organizations, to time-stamp the signer and verifier data two approaches are possible:

* under the terms of the contract pre-defined common "trusted" TSA may be used;

* if both organizations run their own time-stamping services, A and B can have the transaction time-stamped by these two time-stamping services. In the latter case, the electronic signature will only be considered as valid, if both time-stamps were obtained in due time (i.e., there should not be a long delay between obtaining the two time-stamps). Thus, neither A nor B can repudiate the signing time indicated by their own time-stamping service.

Therefore, A and B do not need to agree on a common "trusted" TSA to get a valid transaction.

It is important to note that signatures may be generated "off-line" and time-stamped at a later time by anyone, e.g., by the signer or any recipient interested in validating the signature. The time-stamp over the signature from the signer can thus be provided by the signer together with the signed document, and /or obtained by the verifier following receipt of the signed document.

The business scenarios may thus dictate that one or more of the long-term signature time-stamping methods describe above be used. This will need to be part of a mutually agreed the Signature Validation Policy with is part of the overall signature policy under which digital signature may be used to support the business relationship between the two parties.

B.4.10 TSA Key Compromise

TSA servers should be built in such a way that once the private signature key is installed, that there is minimal likelihood of compromise over as long as possible period. Thus the validity period for the TSA's keys should be as long as possible.

Both the ES-T and the ES-C contain at least one time stamp over the signer's signature. In order to protect against the compromise of the private signature key used to produce that time-stamp, the Archive validation data can be used when a different Time-Stamping Authority key is involved to produce the additional time-stamp. If it is believed that the TSA key used in providing an earlier time-stamp may ever be compromised (e.g., outside its validity period), then the ES-A should be used. For extremely long periods this may be applied repeatedly using new TSA keys.

B.5 Multiple Signatures

Some electronic signatures may only be valid if they bear more than one signature. This is the case generally when a contract is signed between two parties. The ordering of the signatures may or may not

be important, i.e., one may or may not need to be applied before the other. Several forms of multiple and counter signatures may need to be supported, which fall into two basic categories:

* independent signatures;
* embedded signatures.

Independent signatures are parallel signatures where the ordering of the signatures is not important. The capability to have more than one independent signature over the same data must be provided.

Embedded signatures are applied one after the other and are used where the order the signatures are applied is important. The capability to sign over signed data must be provided.

These forms are described in clause 3.13. All other multiple signature schemes, e.g., a signed document with a countersignature, double countersignatures or multiple signatures, can be reduced to one or more occurrence of the above two cases.

Annex C (informative): Identifiers and roles

C.1 Signer Name Forms

The name used by the signer, held as the subject in the signer's certificate, must uniquely identify the entity. The name must be allocated and verified on registration with the Certification Authority, either directly or indirectly through a Registration Authority, before being issued with a Certificate.

This document places no restrictions on the form of the name. The subject's name may be a distinguished name, as defined in [RFC2459], held in the subject field of the certificate, or any other name form held in the X.509 subjectAltName certificate extension field. In the case that the subject has no distinguished name, the subject name can be an empty sequence and the subjectAltName extension must be critical.

C.2 TSP Name Forms

All TSP name forms (Certification Authorities, Attribute Authorities and Time-Stamping Authorities) must be in the form of a distinguished name held in the subject field of the certificate.

The TSP name form must include the legal jurisdiction (i.e., country) under which it operates and an identification for the organization providing the service.

C.3 Roles and Signer Attributes

Where a signer signs as an individual but wishes to also identify him/herself as acting on behalf of an organization, it may be necessary to provide two independent forms of identification. The first identity, with is directly associated with the signing key identifies him/her as an individual. The second, which is managed independently, identifies that person acting as part of the organization, possibly with a given role.

In this case the first identity is carried in the subject/subjectAltName field of the signer's certificate as described above.

This document supports the following means of providing a second form of identification:

* by placing a secondary name field containing a claimed role in the CMS signed attributes field;

* by placing an attribute certificate containing a certified role in the CMS signed attributes field.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.