

S/MIME Working Group  
INTERNET-DRAFT  
Expires February 2006  
Obsoletes: RFC [3126](#)  
Target Category: Informational

J.Ross(Security and Standards)  
N.Pope(Security and Standards)  
D.Pinkas(Bull)  
August 2005

CMS Advanced Electronic Signatures (CADES)  
<[draft-pinkas-smime-cades-01.txt](#)>

#### Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/ietf/1id-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>

This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

#### 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 3369](#) and [RFC 2634](#), where, when appropriate additional signed and unsigned attributes have been defined. The contents of this Informational RFC amounts to a

transposition of the ETSI TS 101 733 V.1.6.3 (CMS Advanced Electronic Signatures - CAdES) and is technically equivalent to it.

Table of Contents

<a href="#">1. Introduction</a>	6
<a href="#">2. Scope</a>	6
<a href="#">3. Definitions and abbreviations</a>	8
<a href="#">3.1 Definitions</a>	8
<a href="#">3.2 Abbreviations</a>	11
<a href="#">4. Overview</a>	12
<a href="#">4.1 Major parties</a>	12
<a href="#">4.2 Signatures policies</a>	14
<a href="#">4.3 Electronic signature formats</a>	14
<a href="#">4.3.1 CAAdES Basic Electronic Signature (CAAdES-BES)</a>	14
<a href="#">4.3.2 CAAdES Explicit Policy Electronic Signatures (CAAdES-EPES)</a>	17
<a href="#">4.4 Electronic signature formats with validation data</a>	18
<a href="#">4.4.1 Electronic Signature with Time (CAAdES-T)</a>	19
<a href="#">4.4.2 ES with Complete validation data references (CAAdES-C)</a>	20
<a href="#">4.4.3 Extended electronic signature formats</a>	22
<a href="#">4.4.4 Archival Electronic Signature (CAAdES-A)</a>	26
<a href="#">4.5 Arbitration</a>	27
<a href="#">4.6 Validation process</a>	28
<a href="#">5. Electronic signature attributes</a>	29
<a href="#">5.1 General syntax</a>	29
<a href="#">5.2 Data content type</a>	29
<a href="#">5.3 Signed-data content type</a>	29
<a href="#">5.4 SignedData type</a>	30
<a href="#">5.5 EncapsulatedContentInfo type</a>	30
<a href="#">5.6 SignerInfo type</a>	30
<a href="#">5.6.1 Message digest calculation process</a>	31
<a href="#">5.6.2 Message signature generation process</a>	31
<a href="#">5.6.3 Message signature verification process</a>	31
<a href="#">5.7 Basic ES mandatory present attributes</a>	31
<a href="#">5.7.1 Content type</a>	31
<a href="#">5.7.2 Message digest</a>	31
<a href="#">5.7.3 Signing certificate reference attribute</a>	31
<a href="#">5.8 Additional mandatory attributes for Explicit Policy-based Electronic Signatures</a>	33
<a href="#">5.8.1 Signature policy identifier</a>	33
<a href="#">5.9 CMS imported optional attributes</a>	35
<a href="#">5.9.1 Signing time</a>	35
<a href="#">5.9.2 Countersignature</a>	35
<a href="#">5.10 ESS imported optional attributes</a>	36
<a href="#">5.10.1 Content reference attribute</a>	36

<a href="#">5.10.2</a>	<b>Content identifier attribute</b>	36
<a href="#">5.10.3</a>	<b>Content hints attribute</b>	36

<a href="#"><u>5.11</u></a>	Additional optional attributes defined in the present document	37
<a href="#"><u>5.11.1</u></a>	Commitment type indication attribute	37
<a href="#"><u>5.11.2</u></a>	Signer location attribute	39
<a href="#"><u>5.11.3</u></a>	Signer attributes attribute	40
<a href="#"><u>5.11.4</u></a>	Content time-stamp	40
<a href="#"><u>5.12</u></a>	Support for multiple signatures	41
<a href="#"><u>5.12.1</u></a>	Independent signatures	41
<a href="#"><u>5.12.2</u></a>	Embedded signatures	41
<a href="#"><u>6.</u></a>	Additional Electronic Signature validation attributes	41
<a href="#"><u>6.1</u></a>	Electronic Signature Time-stamped (CADES-T)	43
<a href="#"><u>6.1.1</u></a>	Signature time- stamp attribute definition	43
<a href="#"><u>6.2</u></a>	Complete validation reference data (CADES-C)	44
<a href="#"><u>6.2.1</u></a>	Complete certificate references attribute definition	44
<a href="#"><u>6.2.2</u></a>	Complete Revocation References attribute definition	45
<a href="#"><u>6.2.3</u></a>	Attribute certificate references attribute definition	47
<a href="#"><u>6.2.4</u></a>	Attribute revocation references attribute definition	47
<a href="#"><u>6.3</u></a>	Extended validation data (CADES-X)	48
<a href="#"><u>6.3.1</u></a>	Time-stamped validation data (CADES-X Type 1 or Type 2)	48
<a href="#"><u>6.3.2</u></a>	Long validation data (CADES-X Long, CADES-X Long Type 1 or 2)	48
<a href="#"><u>6.3.3</u></a>	Certificate values attribute definition	49
<a href="#"><u>6.3.4</u></a>	Revocation values attribute definition	50
<a href="#"><u>6.3.5</u></a>	CADES-C time-stamp attribute definition	51
<a href="#"><u>6.3.6</u></a>	Time-stamped certificates and crls references attribute definition	51
<a href="#"><u>6.4</u></a>	Archive validation data	52
<a href="#"><u>6.4.1</u></a>	Archive time-stamp attribute definition	52
<a href="#"><u>7.</u></a>	Other standard data structures	54
<a href="#"><u>7.1</u></a>	Public-key certificate format	54
<a href="#"><u>7.2</u></a>	Certificate revocation list format	54
<a href="#"><u>7.3</u></a>	OCSP response format	54
<a href="#"><u>7.4</u></a>	Time-stamp token format	54
<a href="#"><u>7.5</u></a>	Name and attribute formats	54
<a href="#"><u>7.6</u></a>	Attribute certificate	55
<a href="#"><u>8.</u></a>	Conformance requirements	55
<a href="#"><u>8.1</u></a>	CADES-Basic Electronic Signature (CADES-BES)	56
<a href="#"><u>8.2</u></a>	CADES-Explicit Policy-based Electronic Signature	56
<a href="#"><u>8.3</u></a>	Verification using time-stamping	56
<a href="#"><u>8.4</u></a>	Verification using secure records	57
<a href="#"><u>9.</u></a>	Security considerations	58
<a href="#"><u>9.1</u></a>	Protection of private key	58
<a href="#"><u>9.2</u></a>	Choice of algorithms	58
<a href="#"><u>10.</u></a>	IANA Considerations	58
<a href="#"><u>11.</u></a>	References	58

<a href="#"><u>11.1</u></a> Normative references	58
<a href="#"><u>11.2</u></a> Informative references	59
<a href="#"><u>12.</u></a> Authors' addresses	62

Annex A (normative): ASN.1 definitions	63
<a href="#">A.1</a> Signature format definitions using X.208 ASN.1 syntax	63
<a href="#">A.2</a> Signature format definitions using X.680 ASN.1 syntax	72
Annex B (informative): Extended forms of Electronic Signatures	81
<a href="#">B.1</a> Extended forms of validation data	81
<a href="#">B.1.1</a> CAAdES-X Long	82
<a href="#">B.1.2</a> CAAdES-X Type 1	83
<a href="#">B.1.3</a> CAAdES-X Type 2	84
<a href="#">B.1.4</a> CAAdES-X Long Type 1 and CAAdES-X Long Type 2	85
<a href="#">B.2</a> Timestamp extensions	87
<a href="#">B.3</a> Archive validation data (CAAdES-A)	88
<a href="#">B.4</a> Example validation sequence	90
<a href="#">B.5</a> Additional optional features	95
Annex C (informative):General description	96
<a href="#">C.1</a> The signature policy	96
<a href="#">C.2</a> Signed information	97
<a href="#">C.3</a> Components of an electronic signature	97
<a href="#">C.3.1</a> Reference to the signature policy	97
<a href="#">C.3.2</a> Commitment type indication	98
<a href="#">C.3.3</a> Certificate identifier from the signer	98
<a href="#">C.3.4</a> Role attributes	99
<a href="#">C.3.4.1</a> Claimed role	99
<a href="#">C.3.4.2</a> Certified role	100
<a href="#">C.3.5</a> Signer location	100
<a href="#">C.3.6</a> Signing time	100
<a href="#">C.3.7</a> Content format	101
<a href="#">C.3.8</a> Content cross referencing	101
<a href="#">C.4</a> Components of validation data	101
<a href="#">C.4.1</a> Revocation status information	101
<a href="#">C.4.1.1</a> CRL information	102
<a href="#">C.4.1.2</a> OCSP information	102
<a href="#">C.4.2</a> Certification path	103
<a href="#">C.4.3</a> Time-stamping for long life of signatures	103
<a href="#">C.4.4</a> Time-stamping for long life of signature before CA key compromises	104
<a href="#">C.4.4.1</a> Time-stamping the ES with complete validation data	105
<a href="#">C.4.4.2</a> Time-stamping certificates and revocation information references	106
<a href="#">C.4.5</a> Time-stamping for archive of signature	107
<a href="#">C.4.6</a> Reference to additional data	108
<a href="#">C.4.7</a> Time-stamping for mutual recognition	108
<a href="#">C.4.8</a> TSA key compromise	109
<a href="#">C.5</a> Multiple signatures	109
Annex D (informative):Data protocols to interoperate with TSPs	110
<a href="#">D.1</a> Operational protocols	110
<a href="#">D.1.1</a> Certificate retrieval	110
<a href="#">D.1.2</a> CRL retrieval	110

<a href="#"><u>D.1.3</u></a> OnLine certificate status	110
<a href="#"><u>D.1.4</u></a> Time-stamping	110
<a href="#"><u>D.2</u></a> Management protocols	110
<a href="#"><u>D.2.1</u></a> Request for certificate revocation	110



Annex E (informative): Guidance on naming	111
<a href="#">E.1</a> Allocation of names	111
<a href="#">E.2</a> Providing access to registration information	111
<a href="#">E.3</a> Naming schemes	112
<a href="#">E.3.1</a> Naming schemes for individual citizens	112
<a href="#">E.3.2</a> Naming schemes for employees of an organization	113
Annex F (informative): Example structured contents and MIME	114
<a href="#">F.1</a> General description	114
<a href="#">F.2</a> Header information	114
<a href="#">F.3</a> Content encoding	115
<a href="#">F.4</a> Multi-part content	115
<a href="#">F.5</a> S/MIME	116
Annex G (informative): Relationship to the European Directive And EESSI	119
<a href="#">G.1</a> Introduction	119
<a href="#">G.2</a> Electronic signatures and the directive	119
<a href="#">G.3</a> ETSI electronic signature formats and the directive	120
<a href="#">G.4</a> EESSI standards and classes of electronic signature	120
<a href="#">G.4.1</a> Structure of EESSI standardization	120
<a href="#">G.4.2</a> Classes of electronic signatures	121
<a href="#">G.4.3</a> EESSI classes and the ETSI electronic signature format	121
Annex H (informative): APIs for the generation and verification of electronic signatures tokens	122
<a href="#">H.1</a> Data framing	122
<a href="#">H.2</a> IDUP-GSS-APIs defined by the IETF	123
<a href="#">H.3</a> CORBA security interfaces defined by the OMG	124
Annex I (informative):Cryptographic algorithms	126
<a href="#">I.1</a> Digest algorithms	126
<a href="#">I.1.1</a> SHA-1	126
<a href="#">I.1.2</a> General	126
<a href="#">I.2</a> Digital signature algorithms	127
<a href="#">I.2.1</a> DSA	127
<a href="#">I.2.2</a> RSA	127
<a href="#">I.2.3</a> General	128
Annex J (informative): Changes from the previous version	130
Full Copyright Statement	131
Disclaimer	131
Intellectual Property	131



## **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 ISO/IEC 10181-5) the validity of the signature).

Thus the present document can be used for any transaction between an individual and a company, between two companies, between an individual and a governmental body, etc. The present 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.

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 present 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 [RFC 2119](#) [[STDWORDS](#)]

## **2. Scope**

The scope the present document covers Electronic Signature Formats only. The aspects of Electronic Signature Policies are defined in [RFC 3125](#) and in [TR 102 272](#) (see **informative references**).

The present document defines a number of Electronic Signature Formats, including 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 (repudiates) the validity of the electronic signature.

The present document specifies use of trusted service providers (e.g. Time-Stamping Authorities), 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 the present 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.



The present document includes the concept of signature policies that can be used to establish technical consistency when validating electronic signatures but does not mandate their use.

The present document is based on the use of public key cryptography to produce digital signatures, supported by public key certificates. The present document also specifies the use of time-stamping and time-marking services to prove the validity of a signature long after the normal lifetime of critical elements of an electronic signature. It also, as an option, defines ways to provide very long-term protection against key compromise or weakened algorithms.

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

- [RFC 3852](#) [4] "Cryptographic Message Syntax (CMS)";
- ISO/IEC 9594-8/ITU-T Recommendation X.509 [1]: "Information technology - Open Systems Interconnection - The Directory: Authentication framework";
- [RFC 3280](#) [2] "Internet X.509 Public Key Infrastructure (PKIX) Certificate and CRL Profile";
- [RFC 3161](#) [7] "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)".

NOTE: See [section 11](#) for a full set of references.

The present document describes formats for advanced electronic signatures using ASN.1 (Abstract Syntax Notation 1). These formats are based on CMS (Cryptographic Message Syntax) defined in [RFC 3852](#) [4]. These electronic signatures are thus called CAAdES, for "CMS Advanced Electronic Signatures".

Another document, TS 101 903 (see informative references), describes formats for XML advanced electronic signatures (XAdES) built on XMLDSIG.

In addition, the present document identifies other documents that define formats for Public Key Certificates, Attribute Certificates, Certificate Revocation Lists and supporting protocols, including, protocols for use of trusted third parties to support the operation of electronic signature creation and validation.



Informative annexes include:

- illustrations of extended forms of extended Electronic Signatures formats that protect against various vulnerabilities and examples of validation processes;
- descriptions and explanations of some of the concepts used in the present document. giving a rational for normative parts of the present document;
- information on protocols to interoperate with Trusted Service Providers;
- information on security considerations;
- an example structured content and MIME;
- the relationship between the present document and the directive on electronic signature and associated standardization initiatives;
- APIs to support the generation and the verification of electronic signatures;
- cryptographic algorithms that may be used;
- guidance on naming.
- changes from the previous version.

### **3 Definitions and abbreviations**

#### **3.1 Definitions**

For the purposes of the present document, the following terms and definitions apply:

Arbitrator: arbitrator entity may be used to arbitrate a dispute between a signer and verifier when there is a disagreement on the validity of a digital signature.

Attribute Authority (AA): authority which assigns privileges by issuing attribute certificates.

Authority certificate: certificate issued to an authority (e.g. either to a certification authority or to an attribute authority).

Attribute Authority Revocation List (AARL): revocation list containing

a list of references to certificates issued to AAs, that are no longer considered valid by the issuing authority.



Attribute Certificate Revocation List (ACRL): revocation list containing a list of references to attribute certificates that are no longer considered valid by the issuing authority.

Certification Authority Revocation List (CARL): revocation list containing a list of public-key certificates issued to certification authorities, that are no longer considered valid by the certificate issuer.

Certification Authority (CA): authority trusted by one or more users to create and assign public key certificates, optionally the certification authority may create the users' keys.

NOTE: See ITU-T Recommendation X.509 [[1](#)].

Certificate Revocation List (CRL): signed list indicating a set of public key certificates that are no longer considered valid by the certificate issuer.

Digital signature: data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery, e.g. by the recipient.

NOTE: See ISO 7498-2 (see informative references).

Electronic signature: data in electronic form which are attached to or logically associated with other electronic data and which serve as a method of authentication.

NOTE: See Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

Enhanced electronic signatures: electronic signatures enhanced by complementing the baseline requirements with additional data, such as time stamp tokens and certificate revocation data, to address commonly recognized threats.

Explicit Policy-based Electronic Signature (EPES): an electronic signature where the signature policy is explicitly specified that shall be used to validate it.

grace period: time period which permits the certificate revocation information to propagate through the revocation process to relying parties.

Initial verification: a process performed by a verifier done after an electronic signature is generated in order to capture additional

information that could make it valid for long term verification.

Public Key Certificate (PKC): public keys of a user, together with some other information, rendered unforgeable by encipherment with the private key of the certification authority which issued it.

NOTE: See ITU-T Recommendation X.509 [[1](#)].

Rivest-Shamir-Adleman (RSA): asymmetric cryptography algorithm based on the difficulty to factorize very large numbers, using a key pair: a private key and a public key.

Signature policy: set of rules for the creation and validation of an electronic signature, that defines the technical and procedural requirements for electronic signature creation and validation, in order to meet a particular business need, and under which the signature can be determined to be valid.

Signature policy issuer: entity that defines and issues a signature policy.

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

Signer: entity that creates an electronic signature.

Subsequent Verification: a process performed by a verifier to assess the signature validity.

NOTE: It may be done even years after the electronic signature was produced by the signer and completed by the Initial Verification and it might not need to capture more data than those captured at the time of initial verification.

Time-Stamp token: data object that binds a representation of a datum to a particular time, thus establishing evidence that the datum existed before that time.

Time-Mark: information in an audit trail from a Trusted Service Provider that binds a representation of a datum to a particular time, thus establishing evidence that the datum existed before that time.

Time-Marking Authority: trusted third party that creates records in an audit trail in order to indicate that a datum existed before a particular point in time.

Time-Stamping Authority (TSA): trusted third party that creates time-stamp tokens in order to indicate that a datum existed at a particular

point in time.

Time-Stamping Unit (TSU): set of hardware and software which is managed as a unit and has a single time-stamp token signing key active at a time.

Trusted Service Provider (TSP): entity that helps to build trust relationships by making available or providing some information upon request.

Validation data: additional data that may be used by a verifier of electronic signatures to determine the signature is valid.

Valid electronic signature: electronic signature which passes validation.

Verifier: entity that verifies evidence.

NOTE 1: See ISO/IEC 13888-1 (see informative references).

NOTE 2: Within the context of the present document this is an entity that validates an electronic signature.

### **3.2 Abbreviations**

For the purposes of the present document, the following abbreviations apply:

AA	Attribute Authority
AARL	Attribute Authority Revocation List
ACRL	Attribute Certificate Revocation List
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation 1
CA	Certification Authority
CAD	Card Accepting Device
CAAdES	CMS Advanced Electronic Signature
CAAdES-A	CAAdES with Archive validation data
CAAdES-BES	CAAdES Basic Electronic Signature
CAAdES-C	CAAdES with Complete validation data
CAAdES-EPES	CAAdES Explicit Policy Electronic Signature
CAAdES-T	CAAdES with Time-stamp
CAAdES-X	CAAdES with eXtended validation data
CARL	Certification Authority Revocation List
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
CWA	CEN Workshop Agreement
DER	Distinguished Encoding Rules (for ASN.1)
DSA	Digital Signature Algorithm
EDIFACT	Electronic Data Interchange For Administration, Commerce and Transport



EESSI	European Electronic Signature Standardization Initiative
EPES	Explicit Policy-based Electronic Signature
ES	Electronic Signature
ESS	Enhanced Security Services (enhances CMS)
IDL	Interface Definition Language
MIME	Multipurpose Internet Mail Extensions
OCSP	Online Certificate Status Provider
OID	Object Identifier
PKC	Public Key Certificate
PKIX	internet X.509 Public Key Infrastructure
RSA	Rivest-Shamir-Adleman
SHA-1	Secure Hash Algorithm 1
TSA	Time-Stamping Authority
TSP	Trusted Service Provider
TST	Time-Stamp Token
TSU	Time-Stamping Unit
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	eXtended Markup Language
XMLDSIG	XML-Signature Syntax and Processing

## **4 Overview**

The present document defines a number of Electronic Signature (ES) formats that build on CMS ([RFC 3852](#) [4]) by adding signed and unsigned attributes.

This clause provides an introduction to the major parties involved (clause 4.1), the concept of Signature Policies (clause 4.2), provides an overview of the various ES formats (clause 4.3), introduces the concept of validation data and provides an overview of formats that incorporate validation data (clause 4.4), presents relevant considerations on arbitration (clause 4.5) and for the validation process (clause 4.6).

The formal specifications of the attributes are specified in clauses 5 and 6, annexes C and D provide rationale for the definitions of the different ES forms.

### **4.1 Major parties**

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

- the Signer;
- the Verifier;
- Trusted Service Providers (TSP);

- the Arbitrator.



The signer is the entity that 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.

The verifier is the entity that validates the electronic signature, it may be a single entity or multiple entities.

The Trusted Service Providers (TSPs) are one or more entities that help to build trust relationships between the signer and verifier. They support the signer and verifier by means of supporting services including: user certificates, cross-certificates, time-stamp tokens, CRLs, ARLs, OCSP responses. The following TSPs are used to support the functions defined in the present document:

- Certification Authorities;
- Registration Authorities;
- Repository Authorities (e.g. a Directory);
- Time-Stamping Authorities;
- Time-Marking Authorities;
- Signature Policy Issuers.

Certification Authorities provide users with public key certificates and with a revocation service.

Registration Authorities allow the identification and registration of entities before a CA generates certificates.

Repository Authorities publish CRLs issued by CAs, signature policies issued by Signature Policy Issuers and optionally public key certificates.

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

Time-Marking Authorities record that some data was formed before a given trusted time.

Signature Policy Issuers define the signature policies to be used by signers and verifiers.

In some cases the following additional TSPs are needed:

- Attribute Authorities.

Attributes Authorities provide users with attributes linked to public key certificates.

An Arbitrator is an entity that arbitrates in disputes between a signer and a verifier.

## **4.2 Signatures policies**

The present document includes the concept of signature policies that can be used to establish technical consistency when validating electronic signatures.

When a comprehensive signature policy used by the verifier is either explicitly indicated by the signer or implied by the data being signed, then a consistent result can be obtained when validating an electronic signature.

When the signature policy being used by the verifier is neither indicated by the signer nor can be derived from other data, or the signature policy is incomplete then verifiers, including arbitrators, may obtain different results when validating an electronic signature. Therefore, comprehensive signature policies that ensure consistency of signature validation are recommended from both the signers and verifiers point of view.

Further information on signature policies is provided in:

- TR 102 038 (see informative references);
- Clauses 5.8.1, C.1 and C.3.1 of the present document;
- [RFC 3125](#) (see informative references);
- TR 102 272 (see informative references).

## **4.3 Electronic signature formats**

The current clause provides an overview for two forms of CMS advanced electronic signature specified in the present document, namely, the CAAdES Basic Electronic Signature (CAAdES-BES) and the CAAdES Explicit Policy-based Electronic Signature (CAAdES-EPES). Conformance to the present document mandates the signer creates one of these formats.

### **4.3.1 CAAdES Basic Electronic Signature (CAAdES-BES)**

A CAAdES Basic Electronic Signature (CAAdES-BES) in accordance with the present contains:

- The signed user data (e.g. the signer's document) as defined in CMS ([RFC 3852](#) [4]);
- A collection of mandatory signed attributes as defined in CMS ([RFC 3852](#) [4]).and in ESS ([RFC 2634](#) [5]);
- Additional mandatory signed attributes defined in the present document;



- The digital signature value computed on the user data and, when present, on the signed attributes, as defined in CMS ([RFC 3852](#) [4]).

A CAvES Basic Electronic Signature (CAvES-BES) in accordance with the present may contain:

- A collection of additional signed attributes;
- A collection of optional unsigned attributes.

The mandatory signed attributes are:

- Content-type. It is defined in [RFC 3852](#) [4] and specifies that the content type of the ContentInfo is "signed-data". Details are provided in clause 5.7.1;
- Message-digest. It is defined in [RFC 3852](#) [4] and specifies the message digest of the eContent OCTET STRING within encapContentInfo being signed. Details are provided in clause 5.7.2;
- ESS signing-certificate OR other-signing-certificate. The ESS signing-certificate attribute is defined in Enhanced Security Services (ESS), [RFC 2634](#) [5] and only allows for the use of SHA-1 as digest algorithm. The other-signing-certificate attribute one is defined in the present document and allows for the use of any digest algorithm. A CAvES-BES claiming compliance with the present document must include one of them. Clause 5.7.3 provides the details of these attributes. Clause 5.7.3.2 shows the formal specification of other-signing-certificate. Rationale for its inclusion is provided in clause C.3.3.

Optional signed attributes may be added to the CAvES-BES, including optional signed attributes defined in CMS ([RFC 3852](#) [4]), ([RFC 2634](#) [5]) and the present document. Listed below are optional attributes that are defined in clause 5 and have a rationale provided in annex C:

- Signing-time: as defined in CMS ([RFC 3852](#) [4]) indicates the time of the signature as claimed by the signer. Details and short rationale are provided in clause 5.9.1. Clause C.3.6 in provides the rationale.
- Content-hints as defined in ESS ([RFC 2634](#) [5]) provides information that describes the format of the signed content. Clause 5.10.1 provides the specification details. Clause C.3.7 in provides the rationale.



- Content-reference. as defined in ESS ([RFC 2634](#) [5]) can be incorporated as a way to link request and reply messages in an exchange between two parties. Clause 5.10.1 provides the specification details. Clause C.3.8 in provides the rationale.
- Content-identifier. as defined in ESS ([RFC 2634](#) [5]) contains an identifier that may be used later on in the previous content-reference attribute. Clause 5.10.2 provides the specification details. Clause C.3.8 in provides the rationale.
- Commitment-type-indication. This attribute is defined by the present document as a way to indicate the commitment endorsed by the signer when producing the signature. Clause 5.11.1 provides the specification details. Clause C.3.2 in provides the rationale.
- Signer-location. This attribute is defined by the present document. It allows the signer to indicate the place where he has purportedly produced the signature. Clause 5.11.2 provides the specification details. Clause C.3.5 provides the rationale.
- Signer-attributes. This attribute is defined by the present document. It allows a claimed or certified role to be incorporated into the signed information. Clause 5.11.3 provides the specification details. Clause C.3.4 provides the rationale.
- Content-time-stamp. This attribute is defined by the present document. It allows a time-stamp token of the data to be signed to be incorporated into the signed information. It provides proof of the existence of the data before the signature was created. Clause 5.11.4 provides the specification details. Clause C.3.6 provides the rationale.

A CAAdES-BES form can also incorporate instances of unsigned attributes as defined in CMS ([RFC 3852](#) [4]) and ([RFC 2634](#) [5]).

- CounterSignature. as defined in CMS ([RFC 3852](#) [4]). It can be incorporated wherever allowing embedded signatures is a requirement. Clause 5.9.2 provides the specification details. Clause C.5 in annex C provides the rationale.

The structure of the CAAdES-BES is illustrated in figure 1.





```

+-----Elect.Signature (CAAdES-BES)-----+
|+-----+ +-----+ +-----+ +-----+ |
||+-----+ +-----+ +-----+ +-----+ || | | | |
|||Signer's | | Signed | Digital | |
|||Document | |Attributes| Signature | |
|||          | |          |          | |
||+-----+ +-----+ +-----+ +-----+ ||
|+-----+ +-----+ +-----+ +-----+ |
+-----+ +-----+ +-----+ +-----+ +

```

Figure 1: Illustration of a CAAdES-BES

The signer's conformance requirements of a CAAdES-BES are defined in clause 8.1.

NOTE: The CAAdES-BES is the minimum format for an electronic signature to be generated by the signer. On its own, it does not provide enough information for it to be verified in the longer term. For example, revocation information issued by the relevant certificate status information issuer needs to be available for long term validation (see clause 4.4.2).

The CAAdES-BES 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 the present document to the Directive). It provides basic authentication and integrity protection.

The semantics of the signed data of a CAAdES-BES or its context may implicitly indicate a signature policy to the verifier. Specification of the contents of signature policies is outside the scope of the present document.

Further information on signature policies is provided in TR 102 038 (see informative references), [RFC 3125](#) (see informative references) and clauses 5.8.1, C.1 and C.3.1 of the present document.

#### **4.3.2 CAAdES Explicit Policy Electronic Signatures (CAAdES-EPES)**

A CAAdES Explicit Policy-based Electronic Signature (CAAdES-EPES) in accordance with the present document, extends the definition of an electronic signature to conform to the identified signature policy. A CAAdES Explicit Policy-based Electronic Signature (CAAdES-EPES) incorporates a signed attribute (signature-policy-identifier) indicating that a signature policy that is mandatory to use to validate the signature and specifies explicitly the signature policy that shall be used. This signed attribute is protected by the signature. The

signature may also have other signed attributes required to conform to the mandated signature policy.

Clause 5.7.3 provides the details on the specification of signature-policy-identifier attribute. Clause C.1 provides a short rationale. Specification of the contents of signature policies is outside the scope of the present document.

Further information on signature policies is provided in TR 102 038 (see informative references) and clauses 5.8.1, C.1 and C.3.1 of the present document.

The structure of the CAAdES-EPES is illustrated in figure 2.

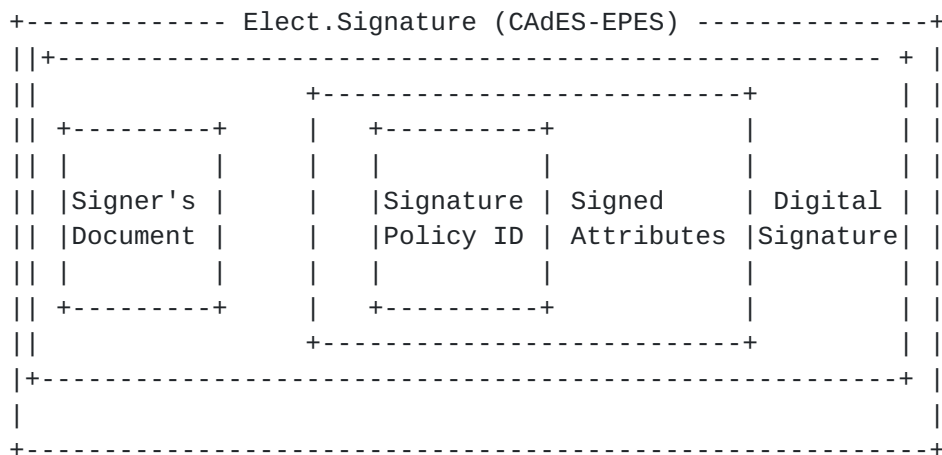


Figure 2: Illustration of a CAAdES-EPES

The signer's conformance requirements of CAAdES-EPES are defined in clause 8.2.

#### **4.4 Electronic signature formats with validation data**

Validation of an electronic signature in accordance with the present document requires additional data needed to validate the electronic signature. This additional data is called validation data; and includes:

- Public Key Certificates (PKCs);
- revocation status information for each PKC;
- trusted time-stamps applied to the digital signature or a time-mark shall be available in an audit log;
- when appropriate, the details of a signature policy to be used to verify the electronic signature.



The validation data may be collected by the signer and/or the verifier. When the signature policy id is present, it shall meet the requirements of the signature policy. Validation data includes CA certificates as well as revocation status information in the form of Certificate Revocation Lists (CRLs) or certificate status information (OCSP) provided by an on-line service. Validation data also includes evidence that the signature was created before a particular point in time this may be either a time-stamp token or time-mark.

The present document defines unsigned attributes able to contain validation data that can be added to CAAdES-BES and CAAdES-EPES leading to electronic signature formats that include validation data. Clauses below summarize these formats and their most relevant characteristics.

#### **4.4.1 Electronic Signature with Time (CAAdES-T)**

Electronic Signature with Time (CAAdES-T) in accordance with the present document is when there exists trusted time associated with the ES.

The trusted time may be provided by:

- the signature-time-stamp as an unsigned attribute added to the ES;
- A time mark of the ES provided by a trusted service provider.

The signature-time-stamp attribute contains a time-stamp token of the electronic signature value. Clause 6.1.1 provides the specification details. Clause C.4.3 in provides the rationale.

A time-mark provided by a Trusted Service would have similar effect to the signature-time-stamp attribute but in this case no attribute is added to the ES as it is the responsibility of the TSP to provide evidence of a time mark when required to do so. The management of time marks is outside the scope of the present document.

Trusted time provides the initial steps towards providing long term validity. Electronic signatures with the time stamp attribute forming the CAAdES-T is illustrated in figure 3.



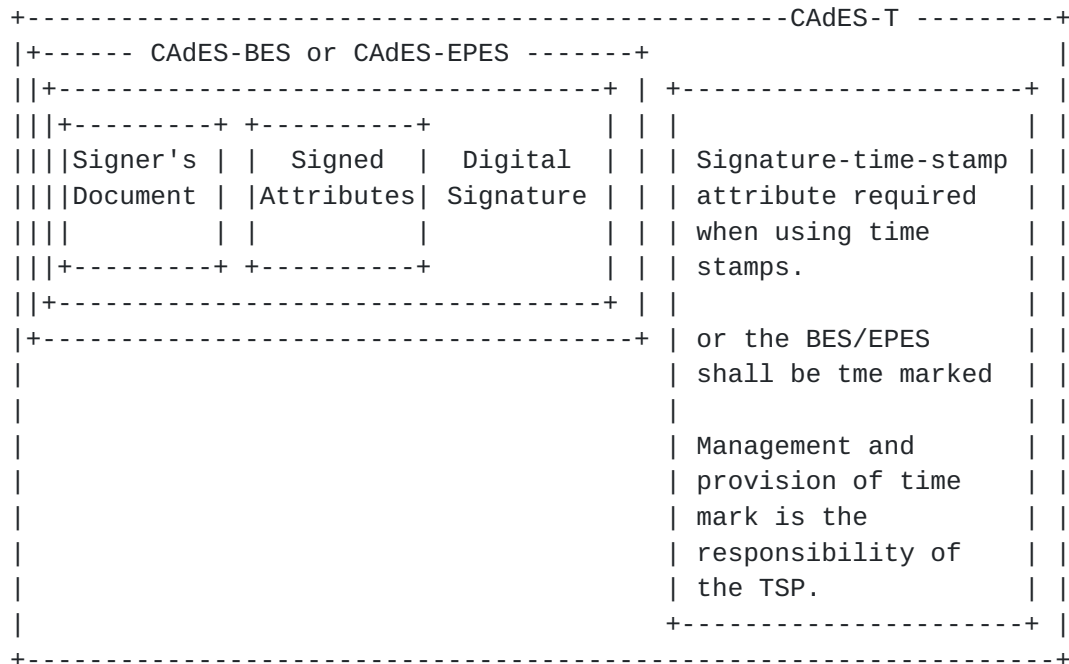


Figure 3: Illustration of CAAdES-T formats

NOTE A time stamp token is added to the CAAdES-BES or CAAdES-EPES as an unsigned attribute.

#### **4.4.2 ES with Complete validation data references (CAAdES-C)**

Electronic Signature with Complete validation data references (CAAdES-C) in accordance with the present document adds to the CAAdES-T the complete-certificate-references and complete-revocation-references attributes as defined by the present document. The complete-certificate-references attribute contain references to all the certificates present in the certification path used for verifying the signature. The complete-revocation-references attribute contains references to the CRLs and/or OCSP responses used for verifying the signature. Clause 6.2 provides the specification details. Storing the references allows the values of the certification path and the CRLs or OCSPs responses to be stored elsewhere, reducing the size of a stored electronic signature format.

Clauses C.4.1 to C.4.2 provide rationale on the usage of validation data and when it is suitable to generate the CAAdES-C form. Electronic signatures with the additional validation data forming the CAAdES-C are illustrated in figure 4.





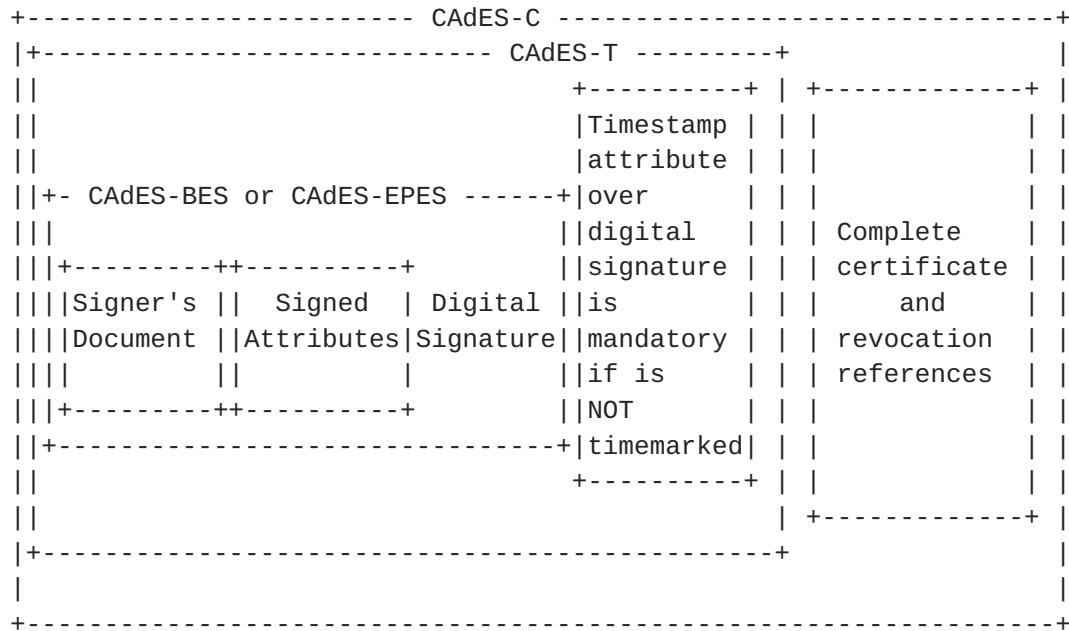


Figure 4: Illustration of CAAdES-C format

- NOTE 1: The complete certificate and revocation references are added to the CAAdES-T as an unsigned attribute.
- NOTE 2: As a minimum, the signer will provide the CAAdES-BES or when indicating that the signature conforms to an explicit signing policy the CAAdES-EPES.
- NOTE 3: To reduce the risk of repudiating signature creation, the trusted time indication needs to be as close as possible to the time the signature was created. The signer or a TSP could provide the CAAdES-T, if not the verifier should create the CAAdES-T on first receipt of an electronic signature because the CAAdES-T provides independent evidence of the existence of the signature prior to the trusted time indication.
- NOTE 4: An CAAdES-T trusted time indications must be created before a certificate has been revoked or expired.
- NOTE 5: The signer and TSP could provide the CAAdES-C, to minimize this risk and when the signer does not provide the CAAdES-C, the verifier should create the CAAdES-C when the required component of revocation and validation data become available, this may require a grace period.
- NOTE 6: A grace period permits certificate revocation information to propagate through the revocation processes. This period could extend from the time an authorized entity requests certificate

revocation, to when the information is available for the  
relying to use. In order to make sure that the certificate  
was not revoked at the time the signature was time-marked or

time-stamped, verifiers should wait until the end of the grace period. A signature policy may define specific values for grace periods. An illustration of a grace period is provided in figure 5.

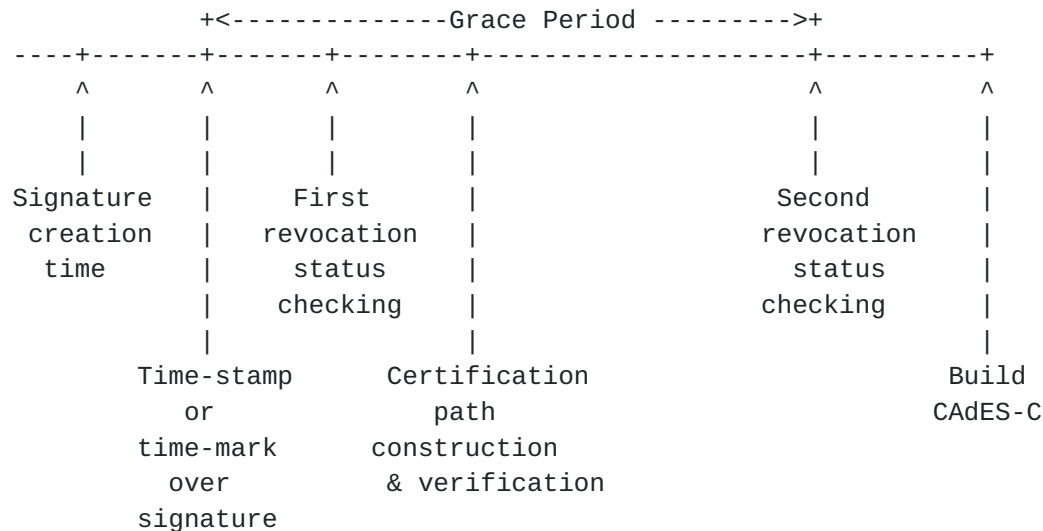


Figure 5: Illustration of a grace period

Figure 5: Illustration of a grace period

NOTE 7: CWA 14171 (see informative references) specifies a signature validation process using CAAdES-T, CAAdES-C and a grace period. Annex B provides example validation processes. Clause C.4 provides additional information about applying grace periods during the validation process.

The verifier's conformance requirements are defined in clause 8.3 for time stamped CAAdES-C and clause 8.4 for time marked CAAdES-C. The present document only defines conformance requirements for the verifier up to an ES with complete validation data (CAAdES-C). This means that none of the extended and archive forms of Electronic Signature as defined in clauses 4.4.3 to 4.4.4) need to be implemented to achieve conformance to the present document.

#### **4.4.3 Extended electronic signature formats**

CAAdES-C can be extended by adding unsigned attributes to the electronic signature. The present document defines various unsigned attributes that are applicable for very long term verification, and for preventing some disaster situations which are discussed in annex C. Annex B provides the details of the various extended formats, all the required unsigned attributes for each type and how they can be used within the electronic signature validation process. The clauses below give an

overview of the various forms of extended signature formats in the present document.

#### 4.4.3.1 Extended Long Electronic Signature (CAAdES-X Long)

Extended Long format (CAAdES-X Long) in accordance with the present document adds to the CAAdES-C format the certificate-values and revocation-values attributes. The first one contains the whole certificate path required for verifying the signature; the second one the CRLs and/OCSP responses required for the validation of the signature. This provides a know repository of certificate and revocation information required to validate an CAAdES-C and prevents such information getting lost. Clauses 6.3.3 and 6.3.4 give specification details. Clause B.1.1 gives details on the production of the format. Clauses C4.1 to C.4.2 provide the rationale.

The structure of the CAAdES-X Long format is illustrated in figure 6.

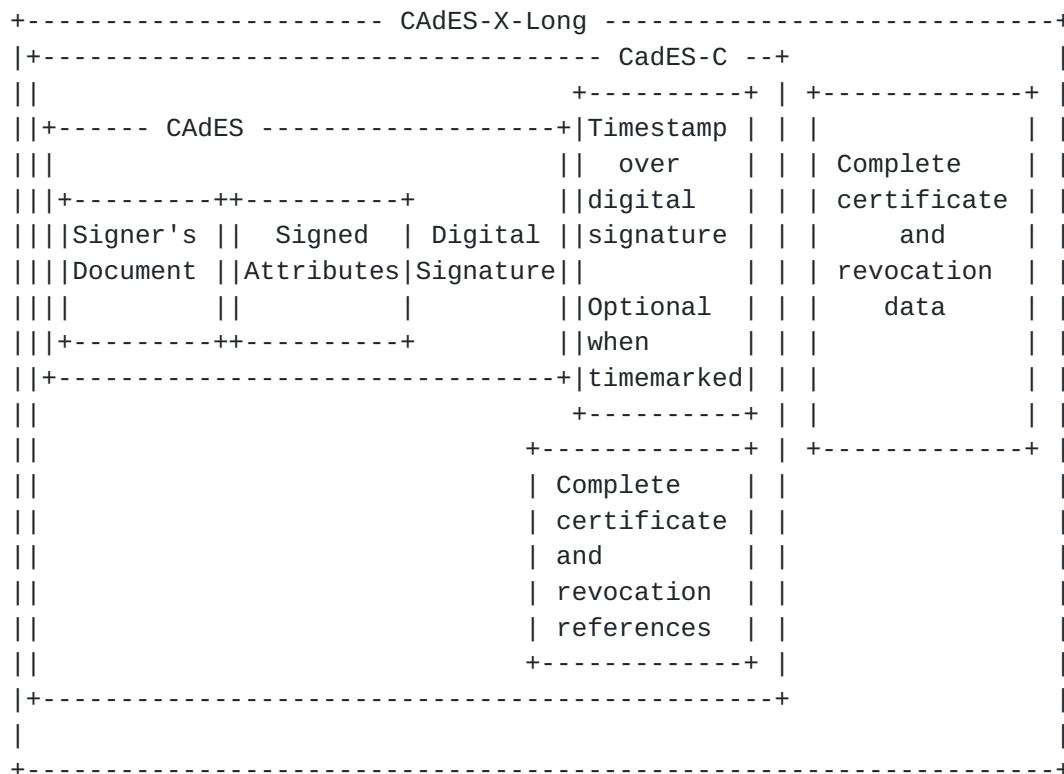


Figure 6: Illustration of CAAdES-X-Long

#### 4.4.3.2 Extended Electronic Signature with Time Type 1 (CAAdES-X Type 1)

Extended format with time type 1 (CAAdES-X Type 1) in accordance with the present document adds to the CAAdES-C format the CAAdES-C-time-stamp attribute, whose content is a time-stamp token on the CAAdES-C itself.

This provides an integrity and trusted time protection over all the elements and references. It may protect the certificates, CRLs and OCSP responses in case of a later compromise of a CA key, CRL key or OCSP issuer key. Clause 6.3.5 provides the specification details.

Clause B.1.2 gives details on the production of the time-stamping process. Clause C.4.4.1 provides the rationale.

The structure of the CAAdES-X Type 1 format is illustrated in figure 7.

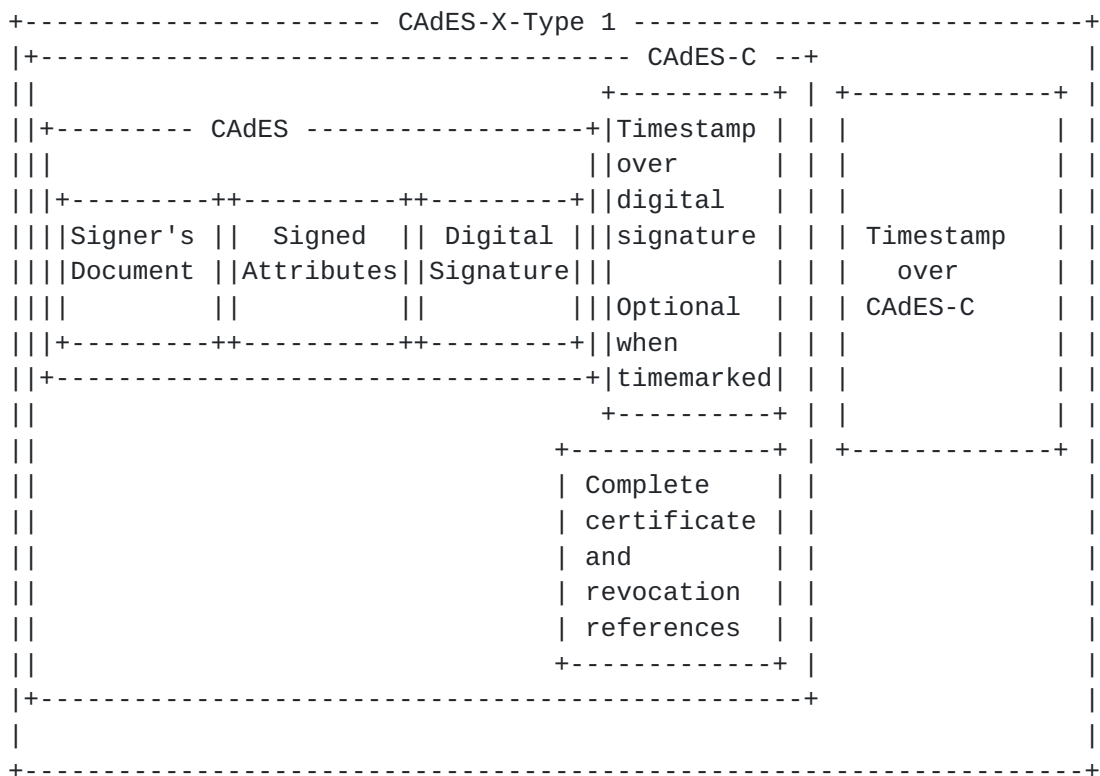


Figure 7: Illustration of CAAdES-X Type 1

#### **4.4.3.3 Extended Electronic Signature with Time Type 2** (CAAdES-X Type 2)

Extended format with time type 2 (CAAdES-X Type 2) in accordance with the present document adds to the CAAdES-C format the CAAdES-C-time-stamped-certs-crls-references attribute, whose content is a time-stamp token on the certification path and revocation information references. This provides an integrity and trusted time protection over all the references.

It may protect the certificates, CRLs and OCSP responses in case of a later compromise of a CA key, CRL key or OCSP issuer key.





Both CAAdES-X Type 1 and CAAdES-X Type 2 counter the same threats and the usage of one or the other depends on the environment. Clause 6.3.5 provides the specification details. Clause B.1.3 gives details on the production of the time-stamping process. Clause C.4.4.2 provides the rationale.

The structure of the CAAdES-X Type 2 format is illustrated in figure 8.

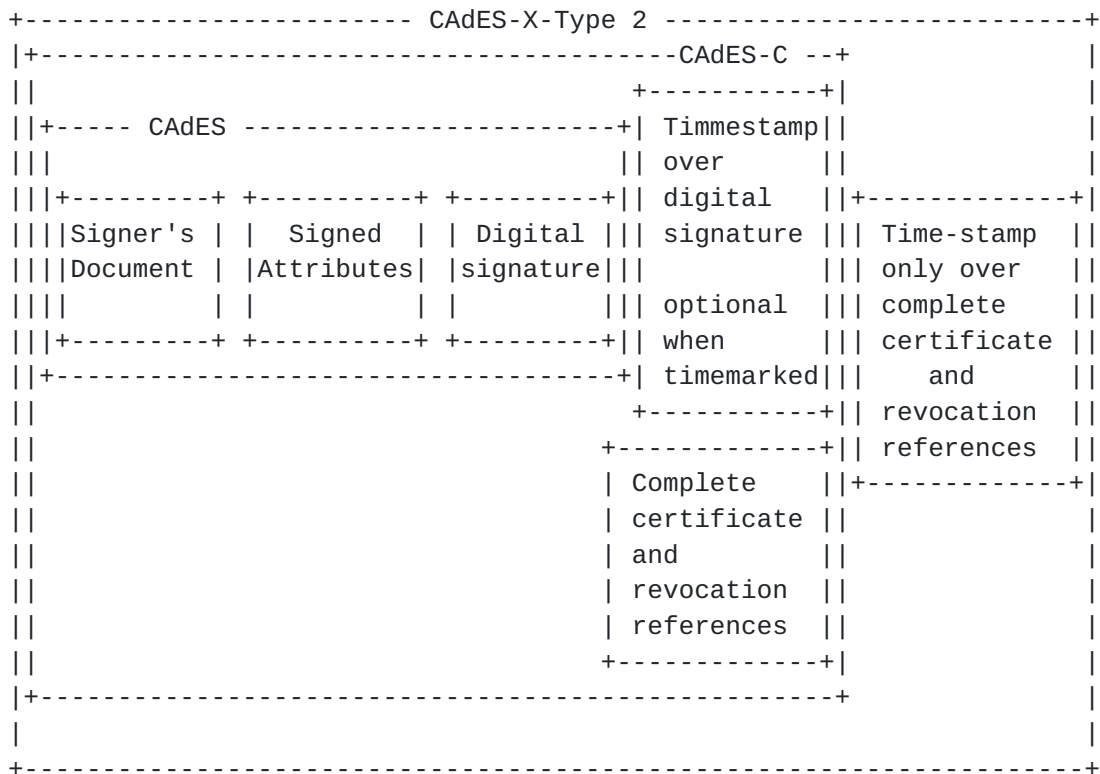


Figure 8: Illustration of CAAdES-X Type 2

#### **4.4.3.4 Extended Long Electronic Signature with Time (CAAdES-X Long Type 1 or 2)**

Extended Long with Time (CAAdES-X Long Type 1 or 2) in accordance with the present document is a combination of CAAdES-X Long and one of the two former types (CAAdES-X Type 1 and CAAdES-X Type 2). Clause B.1.4 gives details on the production of the time-stamping process. Clause C4.8 in annex C provides the rationale.

The structure of the CAAdES-X Long Type 1 and CAAdES-X Long Type 2. format is illustrated in figure 9.



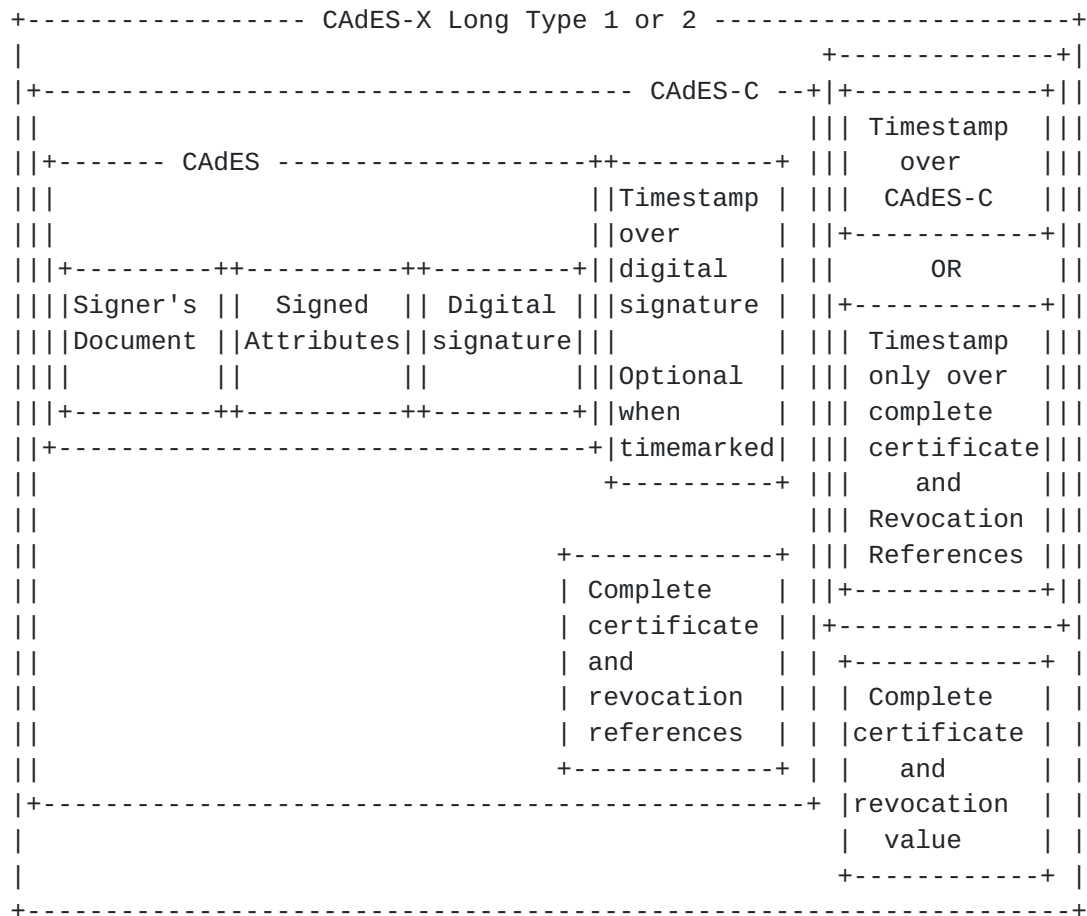


Figure 9: Illustration of CAAdES-X Long Type 1 and CAAdES Long Type 2

#### 4.4.4 Archival Electronic Signature (CAAdES-A)

Archival Form (CAAdES-A) in accordance with the present document builds on a CAAdES-X Long or a CAAdES-X Long Type 1 or 2 by adding one or more archive-time-stamp attributes. This form is used for archival of long-term signatures. Successive time-stamps protect the whole material against vulnerable hashing algorithms or the breaking of the cryptographic material or algorithms. Clause 6.4 contains the specification details. Clauses C.4.5 and C.4.8 provide the rationale.

The structure of the CAAdES-A form is illustrated in figure 10.



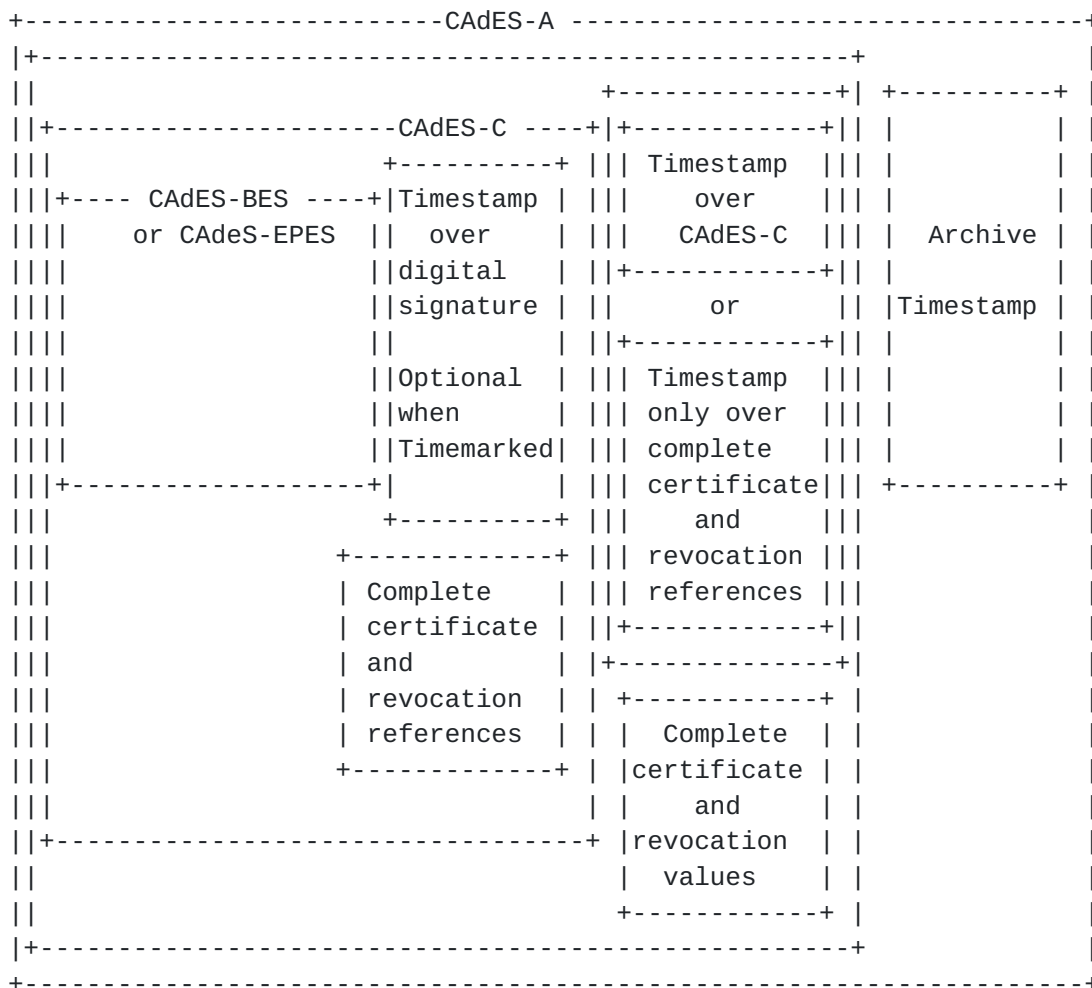


Figure 10: Illustration of CAES-A

TABLE NOTE : Timestamps are timestamp token that may themselves include unsigned attributes required to validate the timestamp token, such as the complete-certificate-references and complete-revocation-references attributes as defined by the present document.

#### 4.5 Arbitration

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

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

- when time-stamping in the CAdES-T is being used, the certificate from the TSU that has issued the time-stamp token in the CAdES-T format is still within its validity period;

- when time-stamping in the CAAdES-T is being used, the certificate from the TSU that has issued the time-stamp token in the CAAdES-T format is not revoked at the time of arbitration;
- when time-marking in the CAAdES-T is being used, a reliable audit trail from the Time-Marking Authority is available for examination regarding the time;
- none of the private keys corresponding to the certificates used to verify the signature chain have ever been compromised;
- the cryptography used at the time the CAAdES-C was built has not been broken at the time the arbitration is performed;
- If the signature policy can be explicit or implicitly identified then an arbitrator is able to determine the rules required to validate the electronic signature.

#### **4.6 Validation process**

The Validation Process validates an electronic signature, the output status of the validation process can be:

- invalid;
- incomplete validation;
- valid.

An Invalid response indicates that either the signature format is incorrect or that the digital signature value fails verification (e.g. the integrity check 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 signature validation status is currently unknown. In the case of incomplete validation, additional information may be made available to the application or user, thus allowing them to decide what to do with the electronic signature. In the case of incomplete validation, the electronic signature may be checked again at some later time when additional information becomes available.

NOTE: For example; an incomplete validation may be because all the required certificates are not available or the grace period is not completed.

A Valid response indicates that the signature has passed verification

and it complies with the signature validation policy.



Example validation sequences are illustrated in annex B.

## **5 Electronic signature attributes**

This clause builds upon the existing Cryptographic Message Syntax (CMS), as defined in [RFC 3852](#) [4], and Enhanced Security Services (ESS), as defined in [RFC 2634](#) [5]. The overall structure of Electronic Signature is as defined in CMS. The Electronic Signature (ES) uses attributes defined in CMS, ESS and the present document. The present document defines 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 the present document. A signature policy MAY mandate that other signed attributes are present.

### **5.1 General syntax**

The general syntax of the ES is as defined in CMS ([RFC 3852](#) [4]).

NOTE: CMS defines content types for id-data, id-signedData, id-envelopedData, id-digestedData, id-encryptedData, and id-authenticatedData. Although CMS permits other documents to define other content types, the ASN.1 type defined should not be a CHOICE type. The present document does not define other content types.

### **5.2 Data content type**

The data content type of the ES is as defined in CMS ([RFC 3852](#) [4]).

NOTE: Requirements to identify encoding types within the content when the ContentType set to id-data are outside the scope of the present document, see annex F for an example of using MIME to identify encoding type.

### **5.3 Signed-data content type**

The signed-data content type of the ES is as defined in CMS ([RFC 3852](#) [4]).

To make sure that the verifier uses the right signer's key, the present document mandates that an unambiguous reference of the signer's certificate is always included in the Signing Certificate signed attribute (see clause 5.7.3).



#### **5.4 SignedData type**

The syntax of the SignedData of the ES is as defined in CMS ([RFC 3852 \[4\]](#)).

The fields of type SignedData have the meanings as defined in CMS ([RFC 3852 \[4\]](#)) **except that:**

- the syntax version number value shall be 3.

The identification of signer's certificate used to create the signature is always signed (see clause 5.7.3). The validation policy may specify requirements for the presence of certain certificates.

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

#### **5.5 EncapsulatedContentInfo type**

The syntax of the EncapsulatedContentInfo type ES is as defined in CMS ([RFC 3852 \[4\]](#)).

For the purpose of long term validation as defined by the present document, it is advisable that either the eContent is present, or the data which is signed is archived in such a way as to preserve 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.

NOTE: The eContent is optional in CMS, this allows the signed data to be encapsulated in the SignData (i.e. Signature + data) alternatively the signed data may be absent from the SignData (i.e. Signature only). It is in the case of signature only that the data which is signed needs to be archived in such a way as to preserve any data encoding.

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

#### **5.6 SignerInfo type**

The syntax of the SignerInfo type ES is as defined in CMS ([RFC 3852 \[4\]](#)).

Per-signer information is represented in the type SignerInfo. In the case of multiple independent signatures (see clause B.5), there is an instance of this field for each signer.

The fields of type SignerInfo have the meanings defined in CMS ([RFC 3852 \[4\]](#)) **except that the signedAttrs field shall contain the following attributes:**



- content-type as defined in clause 5.7.1;
- message-digest as defined in clause 5.7.2;
- signing-certificate as defined in clause 5.7.3.

#### **[5.6.1](#) Message digest calculation process**

The message digest calculation process is as defined in CMS ([RFC 3852](#) [4]).

#### **[5.6.2](#) Message signature generation process**

The input to the message signature generation process is as defined in CMS ([RFC 3852](#) [4]).

#### **[5.6.3](#) Message signature verification process**

The procedures for message signature verification are defined in CMS ([RFC 3852](#) [4]) and enhanced in the present document.

The input to the signature verification process includes the signer's public key which SHALL be verified as correct using the signing certificate reference attribute containing a reference to the signing certificate.

### **[5.7](#) Basic ES mandatory present attributes**

The following attributes SHALL be present with the signed-data defined by the present document. The attributes are defined in CMS ([RFC 3852](#) [4]).

#### **[5.7.1](#) Content type**

The syntax of the content-type attribute type of the ES is as defined in CMS ([RFC 3852](#) [4]).

#### **[5.7.2](#) Message digest**

The syntax of the message-digest attribute type of the ES is as defined in CMS ([RFC 3852](#) [4]).

#### **[5.7.3](#) Signing certificate reference attributes**

The Signing certificate reference attributes are supported by using either the ESS signing-certificate attribute or the other-signing-certificate attribute.

These attributes shall contain a reference to the signer's certificate, they are designed to prevent the simple substitution and re-issue attacks and to allow for a restricted set of certificates to be used in verifying a signature. They have a compact form (much shorter than the full certificate) that allows to a certificate to be unambiguously

identified.

One, and only one, of the following alternative attributes SHALL be present with the signedData defined by the present document.

- The ESS signing-certificate attribute, which is adopted in existing standards, may be used if the SHA-1 hashing algorithm is used.
- The other-signing-certificate attribute shall be used when other hashing algorithms are to be utilized.

The certificate to be used to verify the signature shall be identified in the sequence (i.e. the certificate from the signer) and the sequence shall not be empty. The signature validation policy may mandate other certificates be present that may include all the certificates up to the point of trust.

#### **5.7.3.1 ESS signing certificate attribute definition**

The syntax of the signing-certificate attribute type of the ES is as defined in Enhanced Security Services (ESS), [RFC 2634](#) [5] and further qualified in the present document.

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

The ESS signing-certificate attribute shall be a signed attribute. The encoding of the ESSCertID for this certificate shall include the issuerSerial field.

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

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-attributes attribute as defined in clause 5.8.3.

#### **5.7.3.2 Other signing certificate attribute definition**

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

This attribute shall be used in the same manner as defined above for the ESS signing-certificate attribute.

The following object identifier identifies the other-signing-

certificate attribute:

Pinkas, Pope & Ross

[Page 32]



```
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 other-signing-certificate attribute value has the ASN.1 syntax  
OtherSigningCertificate:

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

## **5.8 Additional mandatory attributes for Explicit Policy-based Electronic Signatures**

### **5.8.1 Signature policy identifier**

The present document mandates that for CAAdES-EPES a reference to the signature policy is included in the signedData. This reference is explicitly identified. A signature policy defines the rules for creation and validation of an electronic signature, is included as a signed attribute with every Explicit Policy-based Electronic Signature. The signature-policy-identifier shall be a signed attribute.

The following object identifier identifies 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  
-- not used in this version}

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

SignaturePolicyImplied ::= NULL

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 optionally contains the identifier of the hash algorithm and the hash of the value of the signature policy.

If the signature policy is defined using ASN.1, then the hash is calculated on the value without the outer type and length fields and the hashing algorithm shall be as specified in the field sigPolicyHash.

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

SigPolicyHash ::= OtherHashAlgAndValue

NOTE: In the previous version of TS 101 733 (i.e. version 1.5.1) sigPolicyHash was mandatory. Implementations requiring to be backward compatible with version 1.5.1 and previous versions of the current document MUST include SigPolicyHash.

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 }
```

The present document specifies the following qualifiers:

- spuri: this contains the web URI or URL reference to the signature policy;
- sp-user-notice: this contains a user notice which should be

displayed whenever the signature is validated.

sigpolicyQualifierIds defined in the present 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)) }

## **5.9 CMS imported optional attributes**

The following attributes MAY be present with the signed-data, the attributes are defined in CMS ([RFC 3852](#) [4]) and are imported into the present document. Where appropriated the attributes are qualified and profiled by the present document.

### **5.9.1 Signing time**

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

Signing-time attribute values for ES have the ASN.1 type SigningTime as defined in CMS ([RFC 3852](#) [4]). This type is further qualified in the present document.

The present document recommends the use of GeneralizedTime.

### **5.9.2 Countersignature**

The counterSignature attribute values for ES have ASN.1 type

CounterSignature as defined in CMS ([RFC 3852](#) [4]).

A counterSignature attribute shall be an unsigned attribute.

#### **5.10 ESS imported optional attributes**

The following attributes MAY be present with the signed-data defined by the present document. The attributes are defined in ESS and are imported into the present document and were appropriate qualified and profiled by the present document.

##### **5.10.1 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 shall be a signed attribute.

Content-reference attribute values for ES have ASN.1 type ContentReference as defined in ESS ([RFC 2634](#) [5]).

The content-reference attribute shall be used as defined in ESS ([RFC 2634](#) [5]) and further qualified in the present document.

##### **5.10.2 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 shall be a signed attribute.

content-identifier attribute type values for of the ES have ASN.1 type ContentIdentifier as defined in ESS ([RFC 2634](#) [5]).

The minimal content-identifier attribute 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.

##### **5.10.3 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 a precise presentation format of the signed data (e.g. text, voice, and video). This attribute SHOULD be present when the signed data is to be presented to human users on verification if the presentation format is not implicit within the data that has been signed.

The syntax of the content-hints attribute type of the ES as defined in

ESS ([RFC 2634](#) [5]).



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

- the contentType 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;
- when the contentType is id-data the contentDescription shall define the presentation format, the format may be defined by MIME types.

When the format of the content is defined by MIME types the following rules apply:

- the contentType shall be id-data as defined in CMS ([RFC 3852](#) [4]);
- the contentDescription shall be used to indicate the encoding of the data in accordance with the rules defined [RFC 2045](#) [6], see annex F for an example structured contents and MIME.

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

NOTE 2: contentDescription is optional in ESS ([RFC 2634](#) [5]). It may be used to complement contentType's defined elsewhere , such definitions are outside the scope of the present document.

### **[5.11](#) Additional optional attributes defined in the present document**

This clause defines a number of attributes that may be used to meet specific requirements.

#### **[5.11.1](#) Commitment type indication attribute**

There may be situations where 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 commitment-type-indication attribute conveys such information.

The commitment-type-indication attribute shall 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 the present document.

The following generic commitment types are defined in the present 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).

#### [5.11.2](#) Signer location attribute

The signer-location attribute 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 [[11](#)]).

The signer-location attribute shall 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 shall be present

countryName	[0]	DirectoryString OPTIONAL, -- As used to name a Country in X.500
localityName	[1]	DirectoryString OPTIONAL, -- As used to name a locality in X.500
postalAddress	[2]	PostalAddress OPTIONAL }

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



### **5.11.3 Signer attributes attribute**

The signer-attributes attribute specifies additional attributes of the signer (e.g. role).

It may be either:

- claimed attributes of the signer;
- certified attributes of the signer.

The signer-attributes attribute shall 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-attributes 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 [RFC 3281](#) : see clause 4.1.

NOTE 1: Only a single signer-attributes can be used

NOTE 2: The claimedAttributes and certifiedAttributes fields are as defined in ITU-T Recommendations X.501 [\[9\]](#) and X.509 [\[1\]](#).

### **5.11.4 Content time-stamp**

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

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

The following object identifier identifies the content-time-stamp 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 of TimeStampToken (as described in [RFC 3161 \[7\]](#)) shall be a hash of the value of eContent field within encapContentInfo in the signedData.

For further information and definition of TimeStampToken see clause 7.4.

NOTE: Content-time-stamp indicates that the signed information was formed before the date included in the Content-time-stamp.

## **[5.12](#) Support for multiple signatures**

### **[5.12.1](#) Independent signatures**

Multiple independent signatures (see clause B.5) are supported by independent SignerInfo from each signer.

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

### **[5.12.2](#) Embedded signatures**

Multiple embedded signatures (see clause C.5) are supported using the countersignature unsigned attribute (see clause 7.1). Each counter signature is carried in Countersignature held as an unsigned attribute to the SignerInfo to which the counter-signature is applied.

## **[6](#) Additional Electronic Signature validation attributes**

This clause specifies attributes that contain different types of validation data. These attributes build on the electronic signature specified in clause 5. This includes:

- Signature-time-stamp applied to the electronic signature value or a Time-Mark in an audit trail. This is defined as the Electronic Signature with Time (CAvES-T);
- complete validation data references which comprises the time-stamp of the signature value (CAvES-T), plus references to all the certificates (complete-certificate-references) and revocation (complete-revocation-references) information used for full validation of the electronic signature. This is defined as the Electronic Signature with Complete data references (CAvES-C).

NOTE 1: Formats for CAvES-T are illustrated in clause 4.4 and the attribute are defined in clause 6.1.1.



NOTE 2: Formats for CAAdES-C are illustrated in clause 4.4. The required attributes for the CAAdES-C signature format are defined in clause 6.2.1 to 6.2.2, optional attributes are defined in clauses 6.2.3 and 6.2.4.

In addition the following optional extended forms of validation data are also defined, see annex B for an overview the eXtended forms of validation data:

- CAAdES-X with time stamp: there are two types of time-stamp used in extended validation data defined by the present document:
  - Type 1(CAAdES-X Type 1): comprises a time-stamp over the ES with complete validation data (CAAdES-C);
  - Type 2 (CAAdES-X Type2): comprises a time-stamp over the certification path references and the revocation information references used to support the CAAdES-C.

NOTE 3: Formats for CAAdES-X Type 1 and CAAdES-X Type 2 are illustrated in clauses B.1.2 and B.1.3 respectively.

- CAAdES-X Long :comprises the complete validation data references (CAAdES-C) plus the actual values of all the certificates and revocation information used in the CAAdES-C.

NOTE 4: Formats for CAAdES-X Long are illustrated in clause B.1.1.

- CAAdES-X Long Type 1 or CAAdES-X Long Type 2: comprises an X-Time-Stamp (Type 1 or Type 2) plus the actual values of all the certificates and revocation information used in the CAAdES-C as per CAAdES-X Long.

This clause also specifies the data structures used in Archive validation data format (CAAdES-A)of eXtended forms:

- Archive form of electronic signature (CAAdES-A) comprises the complete validation data references (CAAdES-C), the certificate and revocation values (as in a CAAdES-X Long ), if present any existing extended electronic signature timestamps (CAAdES-X Type 1 or CAAdES-X Type 2), plus the signed user data and an additional archive time-stamp applied over all that data. An archive time-stamp may be repeatedly applied after long periods to maintain validity when electronic signature and time-stamping 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 6 are unsigned attributes.

NOTE 5: Where multiple signatures are to be supported, as described in clause 5.12, each signature has a separate SignerInfo. Thus, each signature requires its own unsigned attribute values to create CAAdES-T, CAAdES-C, etc.

NOTE 6: the optional attributes of the extended validation data are defined in clauses 6.3 and 6.4.

### **6.1 Electronic Signature Time-stamped (CAAdES-T)**

An Electronic Signature with time-stamp is an electronic signature for which part, but not all, of the additional data required for validation is available (i.e. some certificates and revocation information are available but not all).

The minimum structure time-stamp validation data is:

- the Signature Time-stamp Attribute as defined in clause 6.1.1 over the ES signature value.

#### **6.1.1 Signature time- stamp attribute definition**

The signature-time-stamp attribute is a TimeStampToken computed on the signature value for a specific signer. It is an unsigned attribute. Several instances of this attribute may occur with an electronic signature, from different TSAs.

The following object identifier identifies the signature-time-stamp 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-time-stamp attribute value has ASN.1 type

SignatureTimeStampToken:

SignatureTimeStampToken ::= TimeStampToken

The value of messageImprint field within TimeStampToken shall be a hash of the value of the signature field within SignerInfo for the signedData being time-stamped.

For further information and definition of TimeStampToken see clause 7.4.

NOTE 1: In the case of multiple signatures it is possible to have a TimeStampToken computed for each and all signers, or TimeStampToken computed on one signer's signature and no TimeStampToken on another signer's signature.



NOTE 2: In the case of multiple signatures, several TSTs , issued by different TSAs, may be present within the same signerInfo (see [RFC 3852](#) [4]).

## **6.2 Complete validation reference data (CAAdES-C)**

An electronic signature with complete validation data references (CAAdES-C) is an Electronic Signature for which all the additional data required for validation (i.e. all certificates and revocation information) is available. This form is built on the CAAdES-T form defined above.

As a minimum the complete validation data shall include the following:

- a time, which shall either be a signature-timestamp attribute, as defined in clause 6.1.1, or a time mark operated by a Time-Marking Authority;
- complete-certificate-references, as defined in clause 6.2.1;
- complete-revocation-references , as defined in clause 6.2.2.

### **6.2.1 Complete certificate references attribute definition**

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

NOTE 1: The signer's certificate is referenced in the signing certificate attribute (see clause 5.7.3).

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-references attribute value has the ASN.1 syntax CompleteCertificateRefs.

CompleteCertificateRefs ::= SEQUENCE OF OtherCertID

OtherCertID is defined in clause 5.7.3.2.

The IssuerSerial that shall be present in OtherCertID. The certHash shall match the hash of the certificate referenced.

NOTE 2: Copies of the certificate values may be held using the certificate-values attribute defined in clause 6.3.3.





This attribute MAY include references to the certification chain for any TSUs that provides time-stamp tokens. In this case the unsigned attribute shall be added to the signData of the relevant times tamp token as an unsignedAttrs in the signerInfos field.

### **6.2.2 Complete Revocation References attribute definition**

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

This attribute can be used to illustrate that the verifies has taken due diligence of the available revocation information and then to be able to retrieve that information when stored elsewhere. The following object identifier identifies the complete-revocation-references 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-references attribute value has the ASN.1 syntax CompleteRevocationRefs

```
CompleteRevocationRefs ::= SEQUENCE OF CrlOcspRef
```

```
Cr1OcspRef ::= SEQUENCE {
    Cr1ids      [0]  CRLListID      OPTIONAL,
    Ocspids     [1]  OcspListID     OPTIONAL,
    OtherRev    [2]  OtherRevRefs  OPTIONAL
}
```

CompleteRevocationRefs shall contain one Cr1OcspRef for the signing-certificate, followed by one for each OtherCertID in the CompleteCertificateRefs attribute. The second and subsequent Cr1OcspRef fields shall be in the same order as the OtherCertID to which they relate. At least one of CRLListID or OcspListID or OtherRevRefs should be present for all but the "trusted" CA of the certificate path.

```
CRLListID ::= SEQUENCE {
    crls      SEQUENCE OF Cr1ValidatedID}
```

```
Cr1ValidatedID ::= SEQUENCE {
    crlHash      OtherHash,
    crlIdentifier Cr1Identifier OPTIONAL}
```



```

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

```

```

OcspListID ::= SEQUENCE {
    ocspResponses      SEQUENCE OF OcspResponsesID}

```

```

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

```

```

OcspIdentifier ::= SEQUENCE {
    OcspResponderID    ResponderID,      -- As in OCSP response data
    ProducedAt         GeneralizedTime   -- As in OCSP response data
}

```

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

The `crlIdentifier` is to identify the CRL using the issuer name and the CRL issued time, which shall correspond to the time `thisUpdate` contained in the issued CRL, and if present, the `crlNumber`. The `crlListID` attribute is an unsigned attribute. In the case that the identified CRL is a Delta CRL then references to the set of CRLs to provide a complete revocation list shall be included.

The `OcspIdentifier` is to identify the OCSP response using the issuer name and the time of issue of the OCSP response which shall 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 6.3.4.

```

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 the present document. The definition of the syntax of the other form of revocation information is as identified by OtherRevRefType.

This attribute MAY include the references to the full set of the CRL, ACRL or OCSP responses that have been used to verify the certification chain for any TSUs that provides time-stamp tokens. In this case the unsigned attribute shall be added to the signData of the relevant timestamp token as an unsignedAttrs in the signerInfos field.

### **6.2.3 Attribute certificate references attribute definition**

This attribute is only used when an user attribute certificate is present in the electronic signature.

The attribute-certificate-references attribute is an unsigned attribute. It references the full set of AA certificates that have been used to validate the attribute certificate. Only a single instance of this attribute shall occur with an electronic signature.

id-aa-ets-attrCertificateRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 44 }

The attribute-certificate-references attribute value has the ASN.1 syntax AttributeCertificateRefs.

AttributeCertificateRefs ::= SEQUENCE OF OtherCertID

OtherCertID is defined in clause 5.8.2.

NOTE: Copies of the certificate values may be held using the certificate-values attribute defined in clause 6.3.3.

### **6.2.4 Attribute revocation references attribute definition**

This attribute is only used when a user attribute certificate is present in the electronic signature and when that attribute certificate can be revoked.

The attribute-revocation-references attribute is an unsigned attribute. Only a single instance of this attribute shall occur with an electronic signature. It references the full set of the ACRL or OCSP responses that have been used in the validation of the attribute certificate. This attribute can be used to illustrate that the verifier has taken due diligence of the available revocation information.

The following object identifier identifies the attribute-revocation-references attribute:



id-aa-ets-attrRevocationRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 45 }

The attribute-revocation-references attribute value has the ASN.1 syntax AttributeRevocationRefs.

AttributeRevocationRefs ::= SEQUENCE OF CrlOcspRef

### **6.3 Extended validation data (CAAdES-X)**

This clause specifies a number of optional attributes that are used by extended forms of electronic signatures (see annex B for an overview these forms of validation data).

#### **6.3.1 Time-stamped validation data (CAAdES-X Type 1 or Type 2)**

The extended validation data MAY include one of the following additional attributes, forming a CAAdES-X Time-Stamp validation data (CAAdES-X Type 1 or CAAdES-X Type 2), to provide additional protection against later CA compromise and provide integrity of the validation data used:

- CAAdES-C Time-stamp, as defined in clause 6.3.5 (CAAdES-X Type 1);  
or
- Time-Stamped Certificates and CRLs references, as defined in clause 6.3.6 (CAAdES-X Type 2).

#### **6.3.2 Long validation data (CAAdES-X Long, CAAdES-X Long Type 1 or 2)**

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

- certificate-values as defined in clause 6.3.3;
- revocation-values as defined in clause 6.3.4.

The extended validation data MAY in addition to certificate-values and revocation-values as defined in clauses 6.3.3 and 6.3.4 include one of the following additional attributes, forming an CAAdES-X Long Type 1 or CAAdES-X Long Type 2.

- CAAdES-C Time-stamp, as defined in clause 6.3.3 (CAAdES-X long Type 1); or
- Time-Stamped Certificates and CRLs references, as defined in clause 6.3.4 (CAAdES-X Long Type 2).





The CAAdES-X Long Type 1 or CAAdES-X Long Type 2 provide additional protection against later CA compromise and provide integrity of the validation data used.

NOTE 1: The CAAdES-X Long provides long term proof of a valid electronic signature as long as the CAs are trusted such that these keys cannot be compromised or the cryptography used broken.

NOTE 2: As long as the time stamp data remains valid, the CAAdES-X Long Type 1 and the CAAdES-X Long Type 2 provides the following important property for long standing signatures; that having been found once to be valid, it shall 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.

### **6.3.3 Certificate values attribute definition**

This attribute MAY be used to contain the certificate information required for the following forms of eXtended Electronic Signature: CAAdES-X Long , ES X-Long Type 1 and CAAdES-X Long Type 2, see clause [B.1.1](#) for an illustration of this form of electronic signature.

The certificate-values attribute is an unsigned attribute. Only a single instance of this attribute shall occur with an electronic signature. It holds the values of certificates referenced in the complete-certificate-references attribute.

NOTE: If an attribute certificate is used, it is not provided in this structure but shall be provided by the signer as a signer-attributes attribute (see clause 5.11.3).

The following object identifier identifies the certificate-values 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 clause 7.1 (which is as defined in ITU-T Recommendation X.509 [\[1\]](#)).

This attribute MAY include the certification information for any TSUs that have provided the time-stamp tokens if these certificates are not already included in the TSTs as part of the TSUs signatures. In this

case the unsigned attribute shall be added to the signData of the relevant timestamp token.

#### **6.3.4 Revocation values attribute definition**

This attribute is used to contain the revocation information required for the following forms of eXtended Electronic Signature: CAAdES-X Long, ES X-Long Type 1 and CAAdES-X Long Type 2, see clause B.1.1 for an illustration of this form of electronic signature.

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

```
OtherRevVals ::= SEQUENCE {
    OtherRevValType  OtherRevValType,
    OtherRevVals     ANY DEFINED BY OtherRevValType
}
```

OtherRevValType ::= OBJECT IDENTIFIER

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

CertificateList is defined in clause 7.2 (which as defined in ITU-T Recommendation X.509 [1]).

BasicOCSPResponse is defined in clause 7.3 (which as defined in RFC 2560 [3]).

This attribute MAY include the values of revocation data including CRLs and OCSP for any TSUs that have provided the time-stamp tokens if these certificates are not already included in the TSTs as part of the TSUs

signatures. In this case the unsigned attribute shall be added to the signData of the relevant timestamp token.

### **6.3.5 CAAdES-C time-stamp attribute definition**

This attribute is used to protect against CA key compromise. This attribute is used for the time stamping the complete electronic signature (CAAdES-C). It is used in the following forms of eXtended Electronic Signature; CAAdES-X Type 1 and CAAdES-X Long Type 1, see clause B.1.2 for an illustration of this form of electronic signature.

The CAAdES-C-timestamp attribute is an unsigned attribute. It is a time-stamp token of the hash of the electronic signature and the complete validation data (CAAdES-C). It is a special purpose TimeStampToken Attribute which time-stamps the CAAdES-C. Several instances of this attribute may occur with an electronic signature from different TSAs.

The following object identifier identifies the CAAdES-C-Timestamp 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 CAAdES-C-timestamp attribute value has the ASN.1 syntax ESCTimeStampToken.

```
ESCTimeStampToken ::= TimeStampToken
```

The value of messageImprint field within TimeStampToken shall be a hash of the concatenated values (without the type or length encoding for that value) of the following data objects:

- OCTETSTRING of the SignatureValue field within SignerInfo;
- signature-time-stamp; or a time mark operated by a Time-Marking Authority;
- complete-certificate-references s attribute;
- complete-revocation-references attribute.

For further information and definition of the TimeStampToken see clause 7.4.

### **6.3.6 Time-stamped certificates and crls references attribute definition**

This attribute is used to protect against CA key compromise.

This attribute is used for the time stamping certificate and revocation references. It is used in the following forms of eXtended Electronic

Signature; CAdES-X Type 2 and CAdES-X Long Type 2, see clause B.1.3 for an illustration of this form of electronic signature.

A time-stamped-certs-crls-references attribute is an unsigned attribute. It is a time-stamp token issued for a list of referenced certificates and OCSP responses/CRLs to protect against certain CA compromises. Its syntax is as follows:

The following object identifier identifies the time-stamped-certs-crls-references attribute:

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

- complete-certificate-references attribute;
- complete-revocation-references attribute.

#### **6.4 Archive validation data**

Where an electronic signature is required to last for a very long time, and a the time-stamp token 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 for the archive form of electronic signature (CAAdES-A). This archive time-stamp attribute may be repeatedly applied over a period of time.

##### **6.4.1 Archive time-stamp attribute definition**

The archive-time-stamp attribute is a time-stamp token of many of the elements of the signedData in the electronic signature. If the certificate-values and revocation-values attributes are not present in the CAAdES-BES or CAAdES-EPES, then they shall be added to the electronic signature prior to computing the archive time-stamp token. The archive-time-stamp attribute is an unsigned attribute. Several instances of this attribute may occur with an electronic signature both over time and from different TSUs.

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 TimeStampToken shall be a hash  
of the concatenation of:

- The encapContentInfo element of the SignedData sequence;
- When present, the Certificates and crls elements of the SignedData sequence;
- Together with all data elements in the SignerInfo sequence including all signed and unsigned attributes.

NOTE 1: The SignedData definition is the following:

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

NOTE 2: SignerInfo definition is as follows:

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

Further information and definition of TimeStampToken see clause 7.4.  
The timestamp should be created using stronger algorithms (or longer  
key lengths) than in the original electronic signatures and weak  
algorithm (key length) timestamps.

NOTE 3: This form of ES also provides protection against a TSP key  
Compromise.

The ArchiveTimeStamp will be added as an unsigned attribute in the  
SignerInfo sequence. For the validation of one ArchiveTimeStamp the  
data elements of the SignerInfo must be concatenated excluding all

later ArchivTimeStampToken attributes.

Certificates and revocation information required to validate the ArchiveTimeStamp shall be provided by one of the following methods:

- The TSU provides the information in the SignedData of the timestamp token;
- Adding the complete-certificate-references attribute and the complete-revocation-references attribute of the TSP as an unsigned attribute within TimeStampToken, when the required information is store elsewhere;
- Adding the certificate-values attribute and the revocation-values attribute of the TSP as an unsigned attribute within TimeStampToken, when the required information is store elsewhere.

## **7 Other standard data structures**

### **7.1 Public-key certificate format**

The X.509 v3 certificate basis syntax is defined in ITU-T Recommendation X.509 [[1](#)]. A profile of the X.509 v3 certificate is defined in [RFC 3280](#) [[2](#)], which is being revised. The reader should consult the latest version of this RFC or any RFC that makes it obsolete.

### **7.2 Certificate revocation list format**

The X.509 v2 CRL syntax is defined in ITU-T Recommendation X.509 [[1](#)]. A profile of the X.509 v2 CRL is defined in [RFC 3280](#) [[2](#)], which is being revised.

### **7.3 OCSP response format**

The format of an OCSP token is defined in [RFC 2560](#) [[3](#)].

### **7.4 Time-stamp token format**

The format of a TimeStampToken type is defined in [RFC 3161](#) [[7](#)] and TS [101 861](#) (see informative references).

### **7.5 Name and attribute formats**

The syntax of the naming and other attributes is defined in ITU-T Recommendation X.509 [[1](#)].

The name used by the signer, held as the subject in the signer's certificate, shall be allocated and verified on registration with the Certification Authority, either directly or indirectly through a

Registration Authority, before being issued with a Certificate.

The present document places no restrictions on the form of the name. The subject's name may be a distinguished name, as defined in ITU-T Recommendation X.500 [12], held in the subject field of the certificate, or any other name form held in the subjectAltName certificate extension field as defined in ITU-T Recommendation X.509 [1]. In the case that the subject has no distinguished name, the subject name can be an empty sequence and the subjectAltName extension shall be critical.

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

The TSP name form shall include identifiers for the organization providing the service and the legal jurisdiction (e.g. country) under which it operates.

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 one of the two identities is carried in the subject/subjectAltName field of the signer's certificate as described above.

The present document does not specify the format of signer's attribute that may be included in public key certificates.

NOTE : Signer's attribute may be supported by using a claimed role in the CMS signed attributes field or by placing an attribute certificate containing a certified role in the CMS signed attributes field, see clause 7.6.

## **[7.6](#) Attribute certificate**

The syntax of the AttributeCertificate type is defined in [RFC 3281](#) [13].

## **[8](#). Conformance requirements**

The present document defines conformance requirements for the generation of two forms of basic electronic signature, one of the two forms must be implemented.

The present document defines conformance requirements for the verification of two forms of basic electronic signature, one of the two

forms must be implemented.

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

On verification the inclusion of optional signed and unsigned attributes must be supported only to the extent that the signature is verifiable. The semantics of optional attributes may be unsupported, unless specified otherwise by a signature policy.

### **8.1 CAAdES-Basic Electronic Signature (CAAdES-BES)**

A system supporting CAAdES-BES signers according to the present document shall, 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 3852](#) [4] (see clauses 5.1 and 5.2);
- CMS SignedData as defined in [RFC 3852](#) [4] with version set to 3 and at least one SignerInfo shall be present (see clauses 5.3 to 5.6);
- The following CMS attributes as defined in [RFC 3852](#) [4]:
  - content-type; this shall always be present (see clause 5.7.1);
  - message-digest; this shall always be present (see clause 5.7.2).
- One of following attributes as defined in the present document:
  - signing-certificate: as defined in clause 5.7.3.1;
  - Other-Signing-Certificate as defined in clause 5.7.3.2.

### **8.2 CAAdES-Explicit Policy-based Electronic Signature**

A system supporting Policy-based signers according to the present document shall, at a minimum, support generation of an electronic signature consisting of the previous components defined for the basic signer, plus:

- The following attributes as defined in clause 5.9:
  - signature-policy-identifier; this shall always be present (see clause 5.8.1).

### **8.3 Verification using time-stamping**

A system supporting verifiers according to the present document with

time-stamping facilities shall, at a minimum, support:

- verification of the mandated components of an electronic signature, as defined in clause 8.1.



- signature-time-stamp attribute, as defined in clause 6.1.1.
- complete-certificate-references, attribute as defined in clause 6.2.1.
- complete-revocation-references attribute, as defined in clause 6.2.2.
- Public Key Certificates, as defined in ITU-T Recommendation X.509 [1] (see clause 8.1).
- either of:
  - Certificate Revocation Lists. as defined in ITU-T Recommendation X.509 [1] (see clause 8.2); or
  - on-line Certificate Status Protocol, as defined in [RFC 2560](#) [3] (see clause 8.3).

#### **[8.4](#) Verification 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 clause 8.1;
- complete-certificate-references attribute, as defined in clause 6.2.1;
- complete-revocation-references attribute, as defined in clause 6.2.2;
- a record must be maintained and cannot be undetectable 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] (see clause 8.1);
- either of:
  - Certificate Revocation Lists. as defined in ITU-T Recommendation X.509 [1] (see clause 8.2); or
  - on-line Certificate Status Protocol, as defined in [RFC 2560](#)

[[3](#)] (see clause 8.3).

## **9. Security considerations**

### **9.1 Protection of private key**

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

### **9.2 Choice of algorithms**

Implementers should be aware that cryptographic algorithms become weaker with time. As new cryptanalysis 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.

## **10. IANA Considerations**

Not applicable

## **11. References**

### **11.1 Normative references**

- [1] ITU-T Recommendation X.509 (2000)/ISO/IEC 9594-8 (2001): "Information technology - Open Systems Interconnection - The Directory: Authentication framework".
- [2] IETF [RFC 3280](#) (2002): "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".
- [3] IETF [RFC 2560](#) (1999): "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".
- [4] IETF [RFC 3852](#) (2004): "Cryptographic Message Syntax (CMS)".
- [5] IETF [RFC 2634](#) (1999): "Enhanced Security Services for S/MIME".
- [6] IETF [RFC 2045](#) (1996): "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".
- [7] IETF [RFC 3161](#) (2001): "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)".

- [8] ITU-T Recommendation X.680 (1997): "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation".

- [9] ITU-T Recommendation X.501 (2000)/ISO/IEC 9594-1 (2001): "Information technology - Open Systems Interconnection - Directory models".
- [10] IETF [RFC 3370](#) (2002): "Cryptographic Message Syntax (CMS) Algorithms".
- [11] ITU-T Recommendation F.1: "Operational provisions for the international public telegram service".
- [12] ITU-T Recommendation X.500: "Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services".
- [13] IETF [RFC 3281](#) (2002): "An Internet Attribute Certificate Profile for Authorization".
- [14] ITU-T Recommendation X.208 (1988): "Specification of Abstract Syntax Notation One (ASN.1)".

Referenced documents hereabove which are not found to be publicly available in the expected location might be found at <http://docbox.etsi.org/Reference>.

- [STDWORDS] IETF [RFC 2119](#) Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2](#)

## **[11.2](#) Informative references**

Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

ETSI Standard TS 101 733 V.1.6.3 (2005-06) Electronic Signature Formats. Note: copies of ETSI TS 101 733 can be freely downloaded from the ETSI web site [www.etsi.org](http://www.etsi.org).

IETF [RFC 2246](#) (1999): "The TLS Protocol Version 1.0".

IETF [RFC 2437](#) (1998): "PKCS #1: RSA Cryptography Specifications Version 2.0".

IETF [RFC 2479](#) (1998): "Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP-GSS-API)".

IETF [RFC 2510](#) (1999): "Internet X.509 Public Key Infrastructure

Certificate Management Protocols".

IETF [RFC 2587](#) (1999): "Internet X.509 Public Key Infrastructure LDAPv2 Schema".

IETF [RFC 3125](#) (2000): "Electronic Signature Policies".

IETF [RFC 2559](#) (2003): "Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2".

ETSI TS 101 861: "Time stamping profile".

ETSI TS 101 903: "XML Advanced Electronic Signatures (XAdES)".

ETSI TS 102 023: "Electronic Signatures and Infrastructures (ESI); Policy requirements for time-stamping authorities".

ETSI TS 102 038: "Electronic Signatures and Infrastructures (ESI); XML format for signature policies".

ETSI TR 102 272. "Electronic Signatures and Infrastructures (ESI); ASN.1 format for signature policies".

ISO 7498-2 (1989): "Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture".

ISO/IEC 13888-1 (2004): "IT security techniques - Non-repudiation - Part 1: General".

ISO/IEC 9796-2 (2002): "Information technology - Security techniques - Digital signature schemes giving message recovery - Part 2: Integer factorization based mechanisms".

ISO/IEC 9796-4 (1998): "Digital signature schemes giving message recovery - Part 4: Discrete logarithm based mechanisms".

ISO/IEC 10118-1 (2000): "Information technology - Security techniques - Hash-functions - Part 1: General".

ISO/IEC 10118-2 (2000): "Information technology - Security techniques - Hash-functions - Part 2: Hash-functions using an n-bit block cipher algorithm".

ISO/IEC 10118-3 (2004): "Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions".

ISO/IEC 10118-4 (1998): "Information technology - Security techniques - Hash-functions - Part 4: Hash-functions using modular arithmetic".

ISO/IEC 14888-1 (1998): "Information technology - Security techniques - Digital signatures with appendix - Part 1: General".



ISO/IEC 14888-2 (1999): "Information technology - Security techniques - Digital signatures with appendix - Part 2: Identity-based mechanisms".

ISO/IEC 14888-3 (1998): "Information technology - Security techniques - Digital signatures with appendix - Part 3: Certificate-based mechanisms".

ISO/IEC 15946-2 (2002): "Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 2: Digital signatures".

ISO/IEC 15946-3 (2002): "Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 3: Key establishment".

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

ITU-T Recommendation X.690 (2002): "Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)".

CWA 14171 CEN Workshop Agreements: "General Guidelines for Electronic Signature Verification".

XMLDSIG: W3C/IETF Recommendation (February 2002): "XML-Signature Syntax and Processing".

ANSI X9.30-1 (1997): "Public Key Cryptography for the Financial Services Industry - Part 1: The Digital Signature Algorithm (DSA)".

ANSI X9.30-2 (1997): "Public Key Cryptography for the Financial Services Industry - Part 2: The Secure Hash Algorithm (SHA-1)".

ANSI X9.31-1 (1997): "Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry - Part 1: The RSA Signature Algorithm".

ANSI X9.31-2 (1996): "Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry - Part 2: Hash Algorithms".

ANSI X9.62 (1998): "Public Key Cryptography for the Financial Services Industry - The Elliptic Curve Digital Signature Algorithm (ECDSA)".

IEEE P1363 (2000): "Standard Specifications for Public-Key

Cryptography".

## **12. Authors' addresses**

Denis Pinkas  
Bull S.A.  
Rue Jean-Jaures  
78340 Les Clayes sous Bois CEDEX  
FRANCE

EMail: Denis.Pinkas@bull.net

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

EMail: pope@secstan.com

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

EMail: ross@secstan.com

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

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>



Annex A (normative): ASN.1 definitions

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

**[A.1](#) Signature format definitions using X.208 ASN.1 syntax**

NOTE: The ASN.1 module defined in clause A.1 using syntax defined in ITU-T Recommendation X.208 [[14](#)] has precedence over that defined in clause A.2 in the case of any conflict.

```
ETS-ElectronicSignatureFormats-ExplicitSyntax88 { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-mod(0)
eSignature-explicit88(28)}
```

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS All

IMPORTS

-- Cryptographic Message Syntax (CMS): [RFC 3852](#)

```
ContentInfo, ContentType, id-data, id-signedData, SignedData,
EncapsulatedContentInfo, SignerInfo, id-contentType,
id-messageDigest, MessageDigest, id-signingTime, SigningTime,
id-countersignature, Countersignature
FROM CryptographicMessageSyntax2004
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
smime(16) modules(0) cms-2004(24) }
```

-- 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 3280](#)

```
Certificate, AlgorithmIdentifier, CertificateList, Name,
DirectoryString, Attribute, 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(18)}
```

```
GeneralNames, GeneralName, PolicyInformation
FROM PKIX1Implicit88
{iso(1) identified-organization(3) dod(6) internet(1) security(5)
mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit (19)}
```

-- Internet Attribute Certificate Profile for Authorization - [RFC 3281](#)

```
AttributeCertificate
FROM PKIXAttributeCertificate {iso(1) identified-organization(3)
dod(6) internet(1) security(5) mechanisms(5) pkix(7)
id-mod(0) id-mod-attribute-cert(12)}
```

-- OCSP - [RFC 2560](#)

```
BasicOCSPResponse, ResponderID
FROM OCSP {iso(1) identified-organization(3) dod(6) internet(1)
security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp(14)}
```

-- Time Stamp Protocol [RFC 3161](#)

```
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 the present document

-- =====

-- S/MIME OID arc used in the present 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 the present 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
- the present document 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) }

-- Basic ES CMS Attributes Defined in the present document
-- =====

-- Mandatory RFC 3852 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 THE PRESENT 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 }

-- Policy ES Attributes Defined in the present document
-- =====

-- Mandatory Basic Electronic Signature Attributes as above,
-- plus in addition.

-- 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
    -- not used in this version
}

SignaturePolicyId ::= SEQUENCE {
    sigPolicyId      SigPolicyId,
    sigPolicyHash    SigPolicyHash OPTIONAL,
    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 shall be present  
    countryName    [0]    DirectoryString OPTIONAL,  
        -- As used to name a Country in X.500  
    localityName   [1]    DirectoryString OPTIONAL,  
        -- As used to name a locality in X.500  
    postalAddress  [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 RFC 3281 : see clause 4.1
```

-- Content Timestamp

```
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
```

-- Signature Timestamp

```
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 OCSF response data  
    producedAt                GeneralizedTime -- As in OCSF 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 OPTIONAL}

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

OtherRevValType ::= OBJECT IDENTIFIER

-- CAAdES-C Timestamp

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 Timestamp

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

-- Attribute certificate references

id-aa-ets-attrCertificateRefs OBJECT IDENTIFIER ::= { iso(1)  
member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)  
smime(16) id-aa(2) 44}

AttributeCertificateRefs ::= SEQUENCE OF OtherCertID

-- Attribute revocation references

id-aa-ets-attrRevocationRefs OBJECT IDENTIFIER ::= { iso(1)  
member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)  
smime(16) id-aa(2) 45}

AttributeRevocationRefs ::= SEQUENCE OF CrlOcspRef

END



## **A.2 Signature format definitions using X.680 ASN.1 syntax**

NOTE: The ASN.1 module defined in clause A.2 has precedence over that defined in clause A.2 using syntax defined in ITU-T Recommendation X.680 (1997) [[8](#)] in the case of any conflict.

```
ETS-ElectronicSignatureFormats-ExplicitSyntax97 { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-mod(0)
eSignature-explicit97(29)}
```

DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS All -

IMPORTS

-- Cryptographic Message Syntax (CMS): [RFC 3852](#)

```
ContentInfo, ContentType, id-data, id-signedData, SignedData,
EncapsulatedContentInfo, SignerInfo,
id-contentType, id-messageDigest, MessageDigest, id-signingTime,
SigningTime, id-countersignature, Countersignature
FROM CryptographicMessageSyntax2004
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
smime(16) modules(0) cms-2004(24) }
```

-- 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 3280](#)

```
Certificate, AlgorithmIdentifier, CertificateList, Name,
DirectoryString, Attribute,
FROM PKIX1Explicit88
{iso(1) identified-organization(3) dod(6) internet(1)
security(5) mechanisms(5) pkix(7) id-mod(0)
id-pkix1-explicit(18)}
```

GeneralNames, GeneralName, PolicyInformation

```
FROM PKIX1Implicit88 {iso(1) identified-organization(3) dod(6)
internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
id-pkix1-implicit(19)}
```



-- Internet Attribute Certificate Profile for Authorization - [RFC 3281](#)

AttributeCertificate

FROM PKIXAttributeCertificate {iso(1) identified-organization(3)  
dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)  
id-mod-attribute-cert(12)}

-- OCSP [RFC 2560](#)

BasicOCSPResponse, ResponderID

FROM OCSP {iso(1) identified-organization(3) dod(6) internet(1)  
security(5) mechanisms(5) pkix(7) id-mod(0) id-mod-ocsp(14)}

-- [RFC 3161](#) Internet X.509 Public Key Infrastructure

-- Time-Stamp Protocol (TSP)

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)}

maxSize

FROM ETS-ElectronicSignaturePolicies-97Syntax { iso(1)  
member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)  
smime(16) id-mod(0) 8}

;

-- S/MIME Object Identifier arcs used in the present document

-- =====

-- S/MIME OID arc used in the present 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 the present 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 the present document 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) }

-- Basic ES Attributes Defined in the present document  
-- =====

-- CMS Attributes Defined in the present document

-- Mandatory [RFC 3852](#) 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 THE PRESENT 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 }

-- Policy ES Attributes Defined in the present document  
-- =====

-- Mandatory Basic Electronic Signature Attributes, plus in addition.  
-- 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
    -- not used in this version
}

SignaturePolicyId ::= SEQUENCE {
    sigPolicyId      SigPolicyId,
    sigPolicyHash     SigPolicyHash OPTIONAL,
    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-spq-ets-unotice SIG-QUALIFIER-TYPE
    SPUserNotice }

pointerToSigPolSpec SIG-POLICY-QUALIFIER ::= {
    SIG-POLICY-QUALIFIER-ID id-spq-ets-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 CommitmentTypeIdIdentifier,
    commitmentTypeQualifier SEQUENCE SIZE (1..MAX) OF
        CommitmentTypeQualifier OPTIONAL}
```

```
CommitmentTypeIdIdentifier ::= 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 shall be present  
countryName [0] DirectoryString{maxSize} OPTIONAL,  
-- As used to name a Country in X.500  
localityName [1] DirectoryString{maxSize} OPTIONAL,  
-- As used to name a locality in X.500  
postalAddress [2] PostalAddress OPTIONAL }

PostalAddress ::= SEQUENCE SIZE(1..6) OF DirectoryString{maxSize}

-- 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 [RFC 3281](#) : see clause 4.1

-- Content Timestamp

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

-- Signature Timestamp

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 OCSF response data  
    producedAt              GeneralizedTime -- As in OCSF response data  
}

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

}

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

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

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

-- CAAdES-C Timestamp

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 Timestamp
```

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

-- Attribute certificate references

id-aa-ets-attrCertificateRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 44 }

AttributeCertificateRefs ::= SEQUENCE OF OtherCertID

-- Attribute revocation references

id-aa-ets-attrRevocationRefs OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 45 }

AttributeRevocationRefs ::= SEQUENCE OF CrlOcspRef

END





## Annex B (informative): Extended forms of Electronic Signatures

Clause 4 provides an overview of the various formats of electronic signatures included in the present document. This annex lists the attributes that need to be present in the various extended electronic signature formats and provide example validation sequences using the extended formats.

### **B.1 Extended forms of validation data**

The complete validation data (CAAdES-C) described in clause 4.3 and illustrated in figure 3 may be extended to create Electronic Signatures with extended validation data. Some Electronic Signatures forms that include extended validation are explained below.

An X-Long electronic signature (CAAdES-X Long) is when the values of the certificates and revocation information are added to the CAAdES-C.

This form of Electronic Signature can be useful when the verifier does not have direct access to the following information:

- 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 CAAdES-C.

In some situations additional time-stamps may be created and added to the Electronic Signatures as additional attributes. For example:

- time-stamping all the validation data as held with the ES (CAAdES-C), this eXtended validation data is called a CAAdES-X Type 1; or
- time-stamping individual reference data as used for complete validation. This form of eXtended validation data is called an CAAdES-X Type 2.

NOTE 1: The advantages/drawbacks for CAAdES-X Type 1 and CAAdES-X Type 2 are discussed in clause C.4.4.

The above time-stamp forms can be useful when it is required to counter the risk that any CA keys used in the certificate chain may be compromised.

A combination of the two formats above may be used. This form of eXtended validation data is called an ES X-Long Type 1 or CAAdES-X Long Type 2. This form of Electronic Signature can be useful when the verifier needs both the values and proof of when the validation data

existed.

NOTE 2: The advantages/drawbacks for CAAdES-X long Type 1 and CAAdES-X long Type 2 are discussed in clause C.4.6.

#### **B.1.1 CAAdES-X Long**

An Electronic Signature with the additional validation data forming the CAAdES-X Long form (CAAdES-X-Long)) is illustrated in figure B.1 and comprises the following:

- CAAdES-BES or CAAdES-EPES as defined in clauses 4.3 , 5.7 or 5.8;
- complete-certificate-references attribute as defined in clause 6.2.1;
- complete-revocation-references attribute as defined in clause 6.2.2.

The following attributes are required if a TSP is not providing a time-mark of the ES:

- signature-time-stamp attribute as defined in clause 6.1.1.

The following attributes are required if the full certificate values and revocation values are not already included in the CAAdES-BES or CAAdES-EPES:

- certificate-values attribute as defined in clause 6.3.3;
- revocation-values attribute, as defined in clause 6.3.4.

If attributes certificates are used then the following attributes may be present:

- attribute-certificate-references attribute defined in clause 6.2.3;
- attribute-revocation-references attribute as defined in clause 6.2.4.

Other unsigned attributes may be present, but are not required.

NOTE: Attribute certificate and revocation references are only present if a user attribute certificate is present in the electronic signature, see clauses 6.2.2 and 6.2.3.



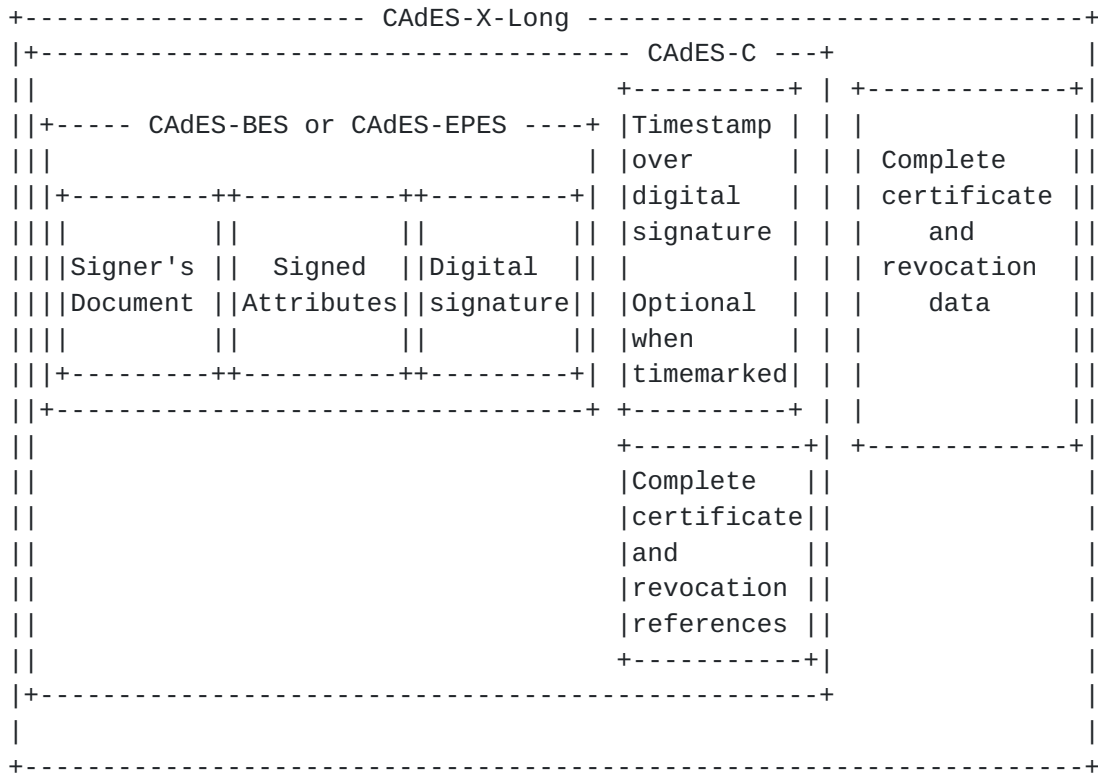


Figure B.1 : Illustration of CAAdES-X-Long

### B.1.2 CAAdES-X Type 1

An Electronic Signature with the additional validation data forming the eXtended Validation Data - Type 1 X is illustrated in figure B.2 and comprises the following:

- the CAAdES-BES or CAAdES-EPES as defined in clauses 4.2, 5.7 or 5.8;
- complete-certificate-references attribute as defined in clause 6.2.1;
- complete-revocation-references attribute as defined in clause 6.2.2;
- CAAdES-C-Timestamp attribute, as defined in clause 6.3.5.

The following attributes are required if a TSP is not providing a time-mark of the ES:

- signature-time-stamp attribute as defined in clause 6.1.1.



If attributes certificates are used then the following attributes may be present:

- attribute-certificate-references attribute defined in clause 6.2.3;
- attribute-revocation-references attribute as defined in clause 6.2.4.

Other unsigned attributes may be present, but are not required.

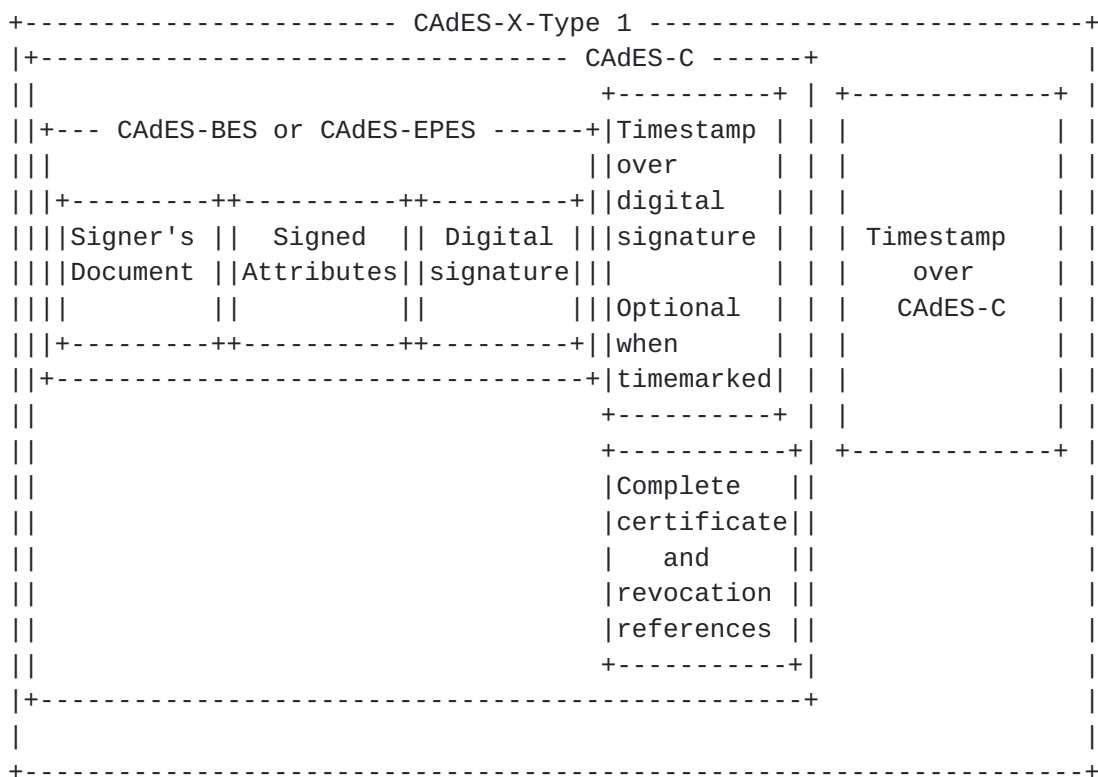


Figure B.2 : Illustration of CAAdES-X Type 1

### **B.1.3 CAAdES-X Type 2**

An Electronic Signature with the additional validation data forming the eXtended Validation Data - Type 2 X is illustrated in figure B.3. and comprises the following:

- CAAdES-BES or CAAdES-EPES as defined in clauses 4.2, 5.7 or 5.8;
- complete-certificate-references attribute as defined in clause 6.2.1;

- complete-revocation-references attribute as defined in clause 6.2.2;





An Electronic Signature with the additional validation data forming the CAdES-X Long Type 1 and CAdES-X Long Type 2 is illustrated in figure B.4 and comprises the following:

- CAAdES-BES or CAAdES-EPES as defined in clauses 4.3, 5.7 or 5.8;
- complete-certificate-references attribute as defined in clause 6.2.1;
- complete-revocation-references attribute as defined in clause 6.2.2;

The following attributes are required if a TSP is not providing a time-mark of the ES:

- signature-time-stamp attribute as defined in clause 6.1.1.

The following attributes are required if the full certificate values and revocation values are not already included in the CAAdES-BES or CAAdES-EPES:

- certificate-values attribute as defined in clause 6.3.3;
- revocation-values attribute, as defined in clause 6.3.4.

If attributes certificates are used then the following attributes may be present:

- attribute-certificate-references attribute defined in clause 6.2.3;
- attribute-revocation-references attribute as defined in clause 6.2.4.

Plus one of the following attributes is required:

- CAAdES-C-Timestamp attribute, as defined in clause 6.3.5;
- time-stamped-certs-crls-references attribute as defined in clause 6.3.6.

Other unsigned attributes may be present, but are not required.



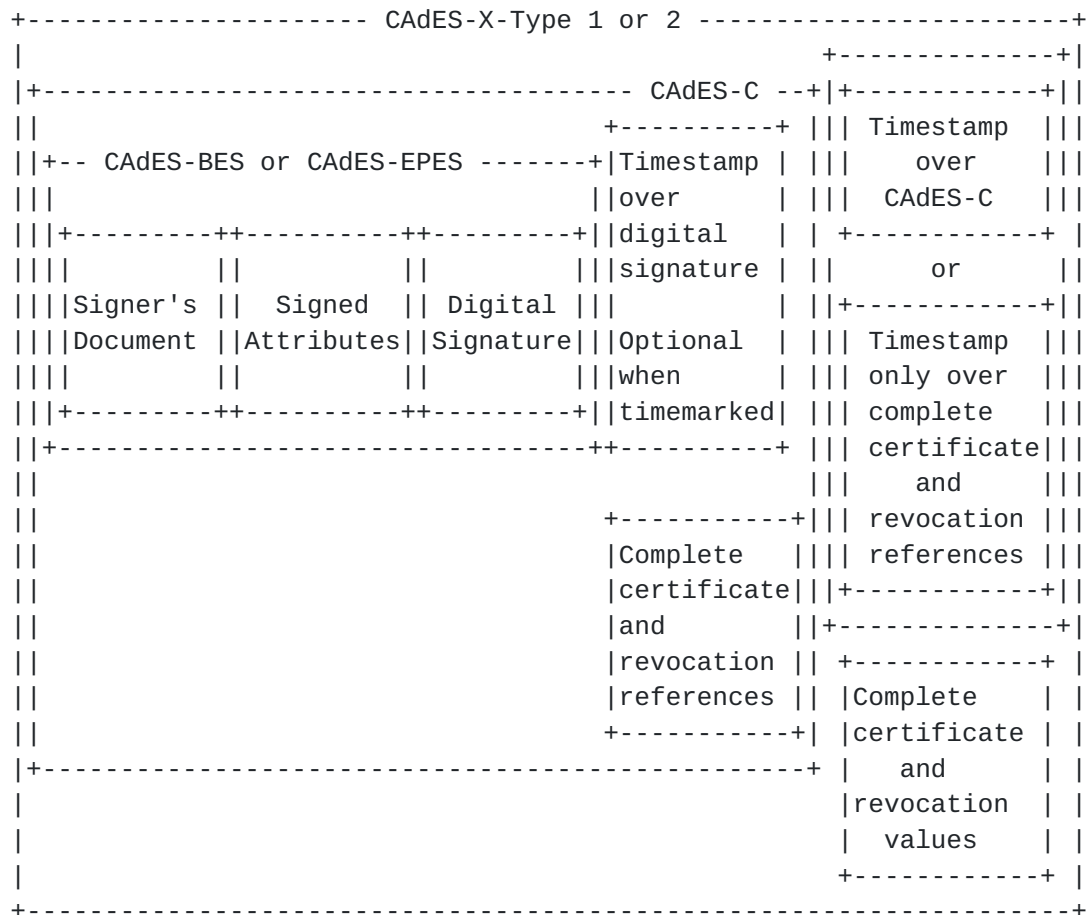


Figure B.4 : Illustration of CAES-X Long Type 1  
and CAES-X Long Type 2

## B.2 Timestamp extensions

Each instance of time-stamp attribute may include as unsigned attributes in the signedData of the timestamp the following attribute related to the TSU:

- complete-certificate-references attribute of the TSU as defined in clause 6.2.1;
- complete-revocation-references attribute of the TSU as defined in clause 6.2.2;
- certificate-values attribute; of the TSU as defined in clause 6.3.3;
- revocation-values attribute, of the TSU as defined in clause 6.3.4.

Other unsigned attributes may be present, but are not required.

**B.3 Archive validation data (CADES-A)**

Before the algorithms, keys and other cryptographic data used at the time the CADES-C was built become weak and the cryptographic functions become vulnerable, or the certificates supporting previous time-stamps expires, the signed data, the CADES-C and any additional information (i.e. any CADES-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 required for the ES Archive format (CADES-A). The Time-stamping process may be repeated every time the protection used to time-stamp a previous CADES-A becomes weak. An CADES-A may thus bear multiple embedded time stamps.

An example of an Electronic Signature (ES), with the additional validation data for the CADES-C and CADES-X forming the CADES-A is illustrated in figure B.5.

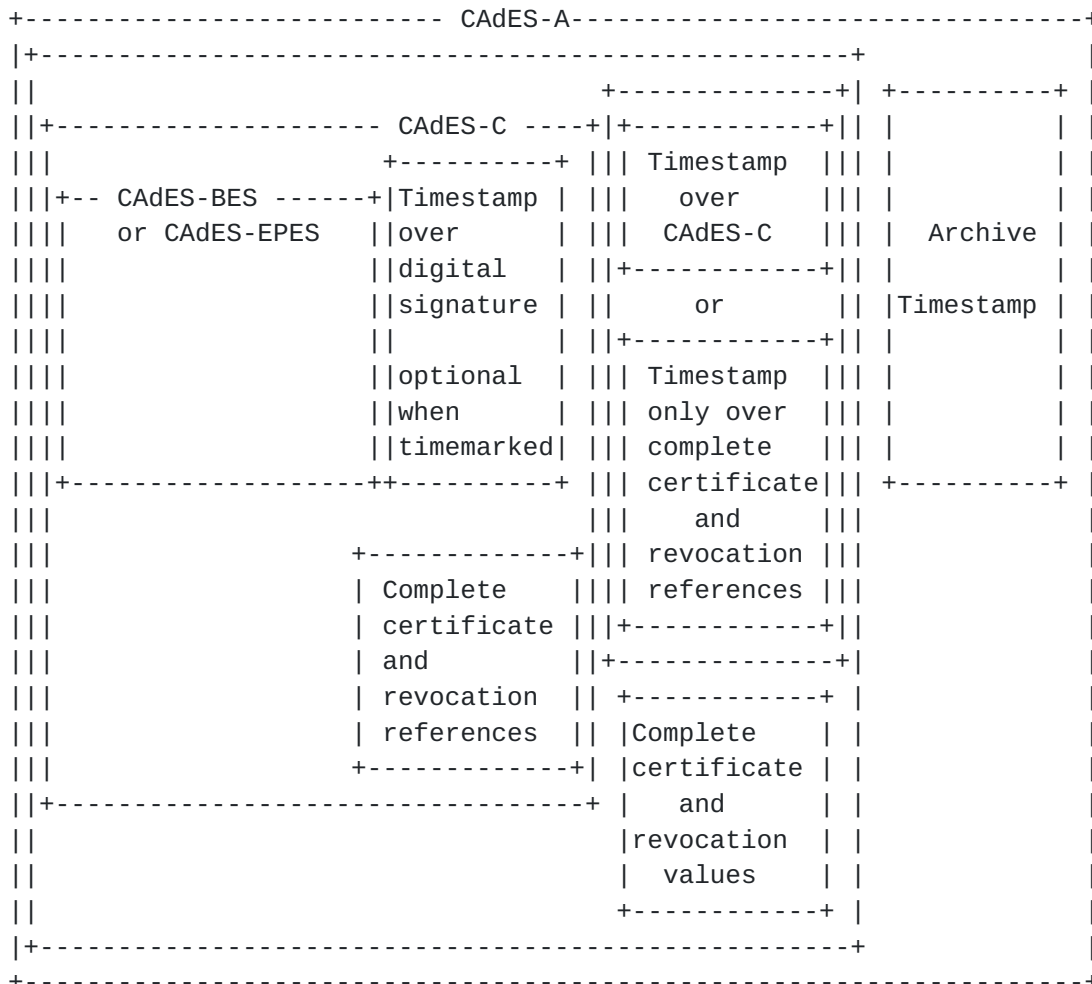


Figure B.5 : Illustration of CAdES-A



The CAAdES-A comprises the following elements:

- the CAAdES-BES or CAAdES-EPES including their signed and unsigned attributes;
- complete-certificate-references attribute as defined in clause 6.2.1;
- complete-revocation-references attribute as defined in clause 6.2.2.

The following attributes are required if a TSP is not providing a time-mark of the ES:

- signature-time-stamp attribute as defined in clause 6.1.1.

If attributes certificates are used then the following attributes may be present:

- attribute-certificate-references attribute defined in clause 6.2.3;
- attribute-revocation-references attribute as defined in clause 6.2.4.

The following attributes are required if the full certificate values and revocation values are not already included in the CAAdES-BES or CAAdES-EPES:

- certificate-values attribute as defined in clause 6.3.3;
- revocation-values attribute as defined in clause 6.3.4.

At least one of the following two attributes is required:

- CAAdES-C-Timestamp attribute as defined in clause 6.3.5;
- time-stamped-certs-crls-references attribute as defined in clause 6.3.6.

The following attribute is required:

- archive-time-stamp attributes defined in clause 6.4.1.

Several instances of archive-time-stamp attribute may occur with an electronic signature both over time and from different TSUs. The time-stamp should be created using stronger algorithms (or longer key lengths) than in the original electronic signatures or time-stamps.

Other unsigned attributes of the ES may be present, but are not required.



The archive timestamp will itself contain the certificate and revocation information required to validate the archive timestamp, this may include the following unsigned attributes:

- complete-certificate-references attribute of the TSU as defined in clause 6.2.1;
- complete-revocation-references attribute of the TSU as defined in clause 6.2.2;
- certificate-values attribute of the TSU as defined in clause 6.3.3;
- revocation-values attribute of the TSU as defined in clause 6.3.4.

Other unsigned attributes may be present, but are not required.

#### **B.4 Example validation sequence**

As described earlier the signer or initial verifier may collect all the additional data that forms the electronic signature. Figure B.6, and subsequent description, describes how the validation process may build up a complete electronic signature over time.



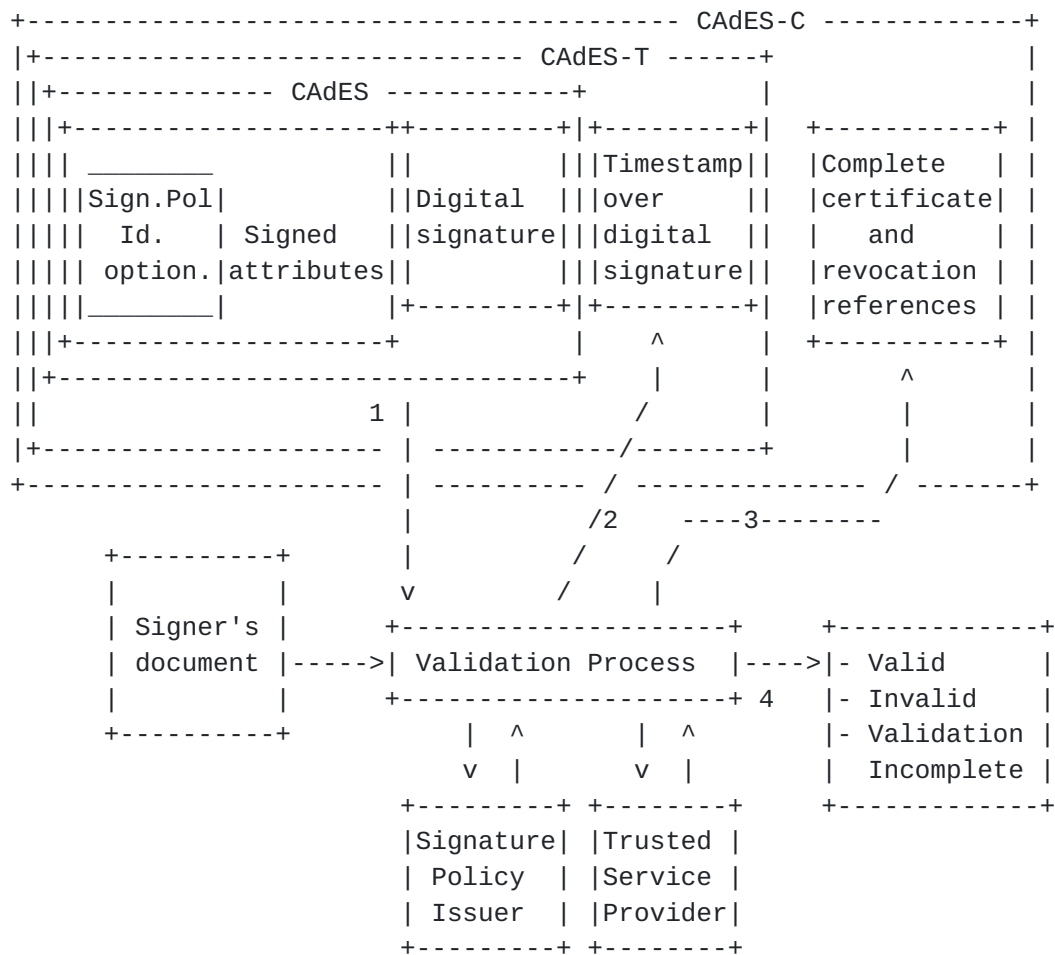


Figure B.6 : Illustration of a CAdES validation sequence

Soon after receiving the Electronic Signature (CADES) from the signer (1), the digital signature value may be checked; the validation process shall 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, using additional data (e.g. certificates, CRL, etc.) provided by trusted service providers. When applicable, the validation process will also need to conform to the requirements specified in a signature policy. If the validation process is validation incomplete, then the output from this stage is the CADES-T.

To ascertain the validity status as Valid or Invalid and communicate that to the user (4) all the additional data required to validate the CAdES-C, must be available (e.g. the complete certificate and revocation information).

Once the data needed to complete validation data references (CAdES-C)

is available then the validation process should:

- obtain all the necessary additional certificate and revocation status information;

- complete 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 CAdES-T and CAdES-C process);
- record the complete certificate and revocation references (3);
- indicate the validity status to the user (4).

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

This is illustrated in figure B.7.

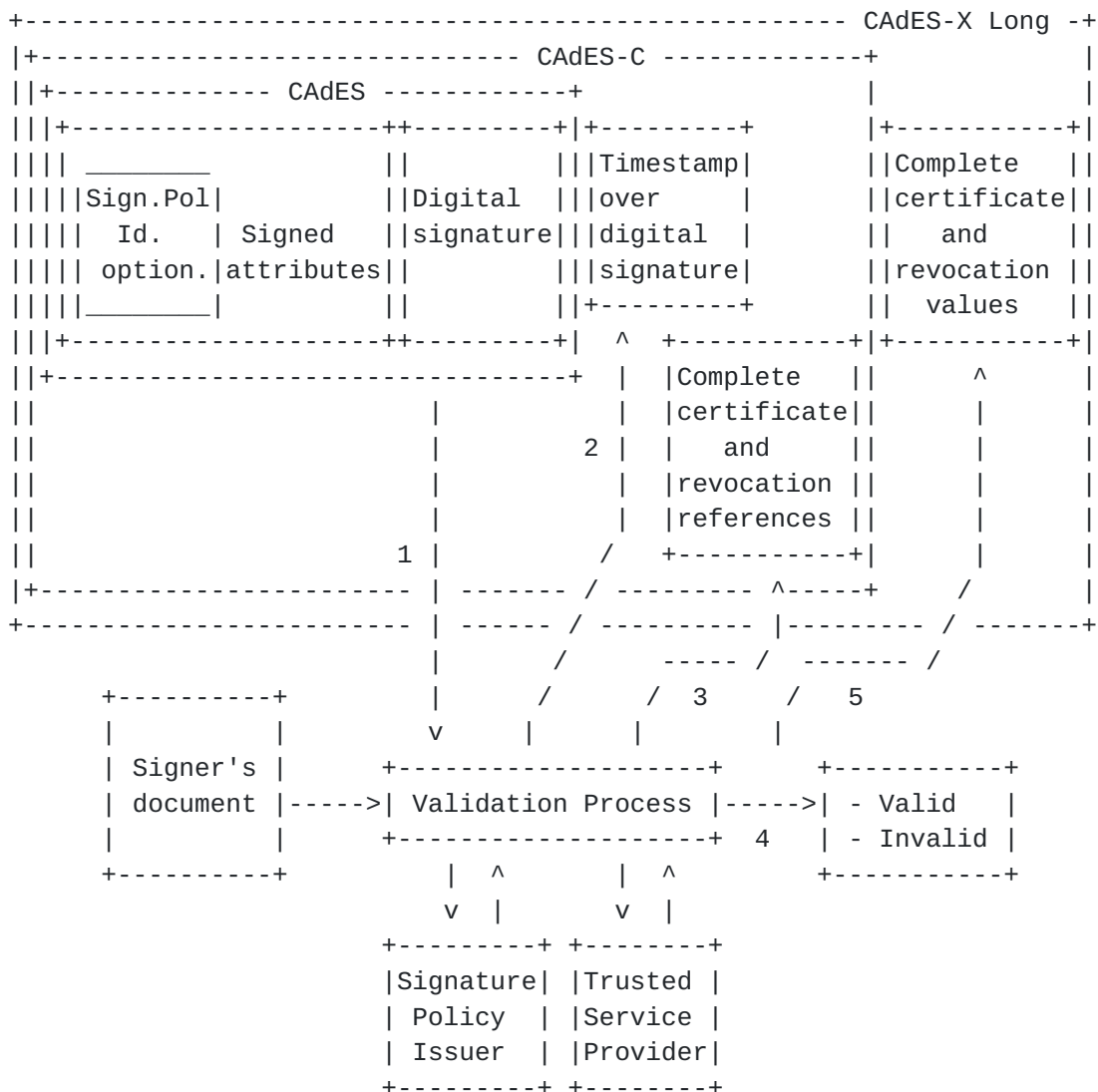


Figure B.7 : Illustration of a CAdES validation sequence  
with CAdES-X Long



When the validation process creates the CADES-C it may also create extended forms of validation data.

A first alternative is to time-stamp all data forming the CADES-X Type 1 [1 \(6\)](#).

This is illustrated in figure B.8.

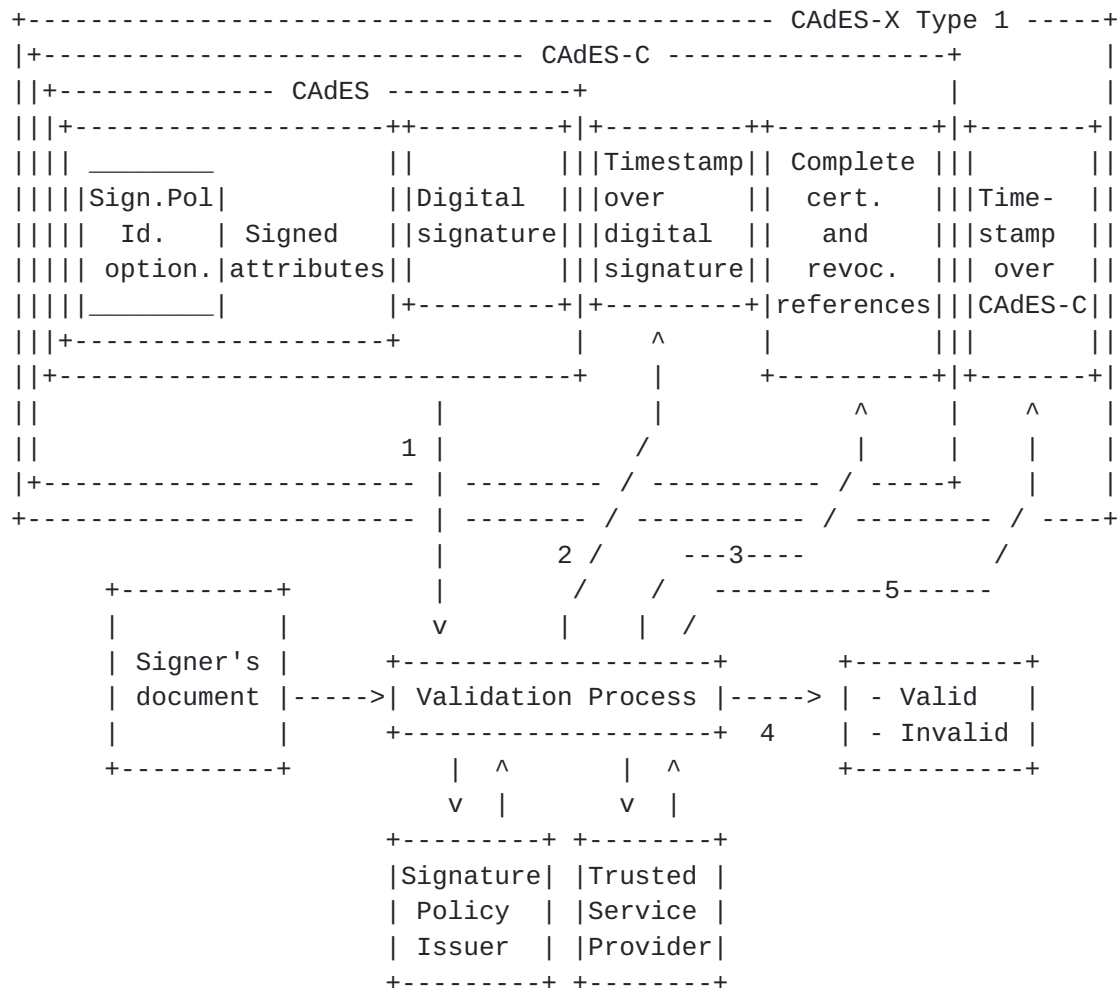


Figure B.8 : Illustration of CADES with eXtended Validation Data  
CADES-X Type 1

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 CADES-X Type 2.

This is illustrated in figure B.9.



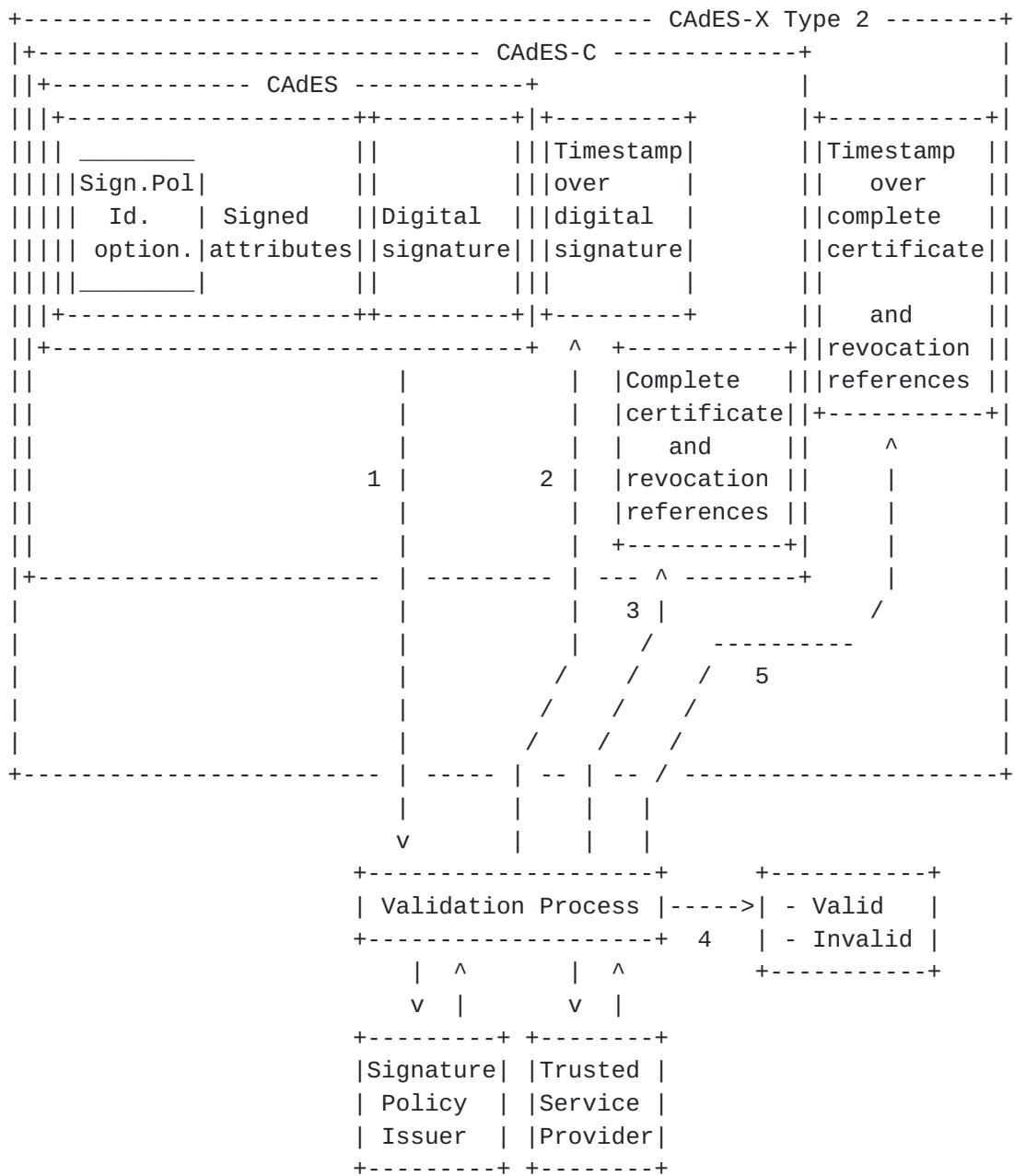


Figure B.9: Illustration of CAdES with eXtended Validation Data  
CAdES-X Type 2

Before the algorithms used in any of electronic signatures become or are likely, to be compromised or rendered vulnerable in the future, it may be 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 (CADES-A) (7).





The present document also defines additional optional features to:

- indicate a commitment type being made by the signer;
- indicate the claimed time when the signature was done;
- indicate the claimed location of the signer;
- indicate the claimed or certified role under which a signature was created;

- support counter signatures;
- support multiple signatures.

## Annex C (informative):General description

This annex explains some of the concepts and provides the rational for normative parts of the present document.

The specification below includes a description why and when the each component of an electronic signature is useful, with a brief description of the vulnerabilities and threats and the manner by which they are countered.

### **C.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 specifies the electronic rules for the creation and validation of the electronic signature also needs to be comprehensively defined and in a computer processable form.

The signature policy thus includes the following:

- rules, which apply to technical validation of a particular signature;
- 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.





For example, the major rules required for technical validation can include: recognized root keys or "top-level certification authorities", acceptable certificate policies (if any), necessary certificate extensions and values (if any), the need for the revocation status for each component of the certification tree, acceptable TSAs (if time-stamp tokens are being used), acceptable organizations for keeping the audit trails with time-marks (if time-marking is being used), acceptable AAs (if any are being used).as well as rules defining the components of the electronic signature that shall be provided by the signer with data required by the verifier when required to provide long term proof.

## **C.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 data, free text or fields from an electronic form (e-form). For example, the Adobe(tm) format "pdf" may be used or the eXtensible Markup Language (XML). Annex D defines how the content may be structured to indicate the type of signed data using MIME.

## **C.3 Components of an electronic signature**

### **C.3.1 Reference to the signature policy**

When two independent parties want to evaluate an electronic signature, it is fundamental that they get the same result. This requirement can be met using comprehensive signature policies that ensure consistency of signature validation. Signature policies can be identified implicitly by the data being signed or they can be explicitly identified using the CAAdES-EPES form of electronic signature, the CAAdES-EPES mandates a consistent signature policy must be used by both the signer and verifier.

By signing over the signature policy identifier in the CAAdES-EPES the signer explicitly indicates that he or she has applied the signature policy in creating the signature.

In order to unambiguously identify the details of an explicit signature policy that is to be used to verify a CAAdES-EPES 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.

In order to unambiguously identify the authority responsible for defining an explicit signature policy the "Signature policy" can be

signed.

### **C.3.2 Commitment type indication**

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 may be subject to signer and verifier agreement, specified as part of the signature policy or registered for generic use across multiple policies.

If a CAvES-EPES electronic signature format is used and the electronic signature includes a commitment type indication other than one of those recognized under the signature policy the signature shall be treated as invalid.

How commitment is indicated using the semantics of the data being signed is outside the scope of the present 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).

### **C.3.3 Certificate identifier from the signer**

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 vulnerable to substitution attacks. 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.

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

#### **C.3.4 Role attributes**

While the name of the signer is important, the position of the signer within a company or an organization of paramount importance as well. Some information (i.e. a contract) may only be valid if signed by a user in a particular role, e.g. a Sales Director. In many cases who 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.

The present document defines two different ways for providing this feature:

- by placing a claimed role name in the CMS signed attributes field;
- by placing an 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 identity certificate. However, it was decided not to follow this approach as it significantly complicates the management of certificates. For example by using separate certificates for signer's identity and roles means new identity keys need not be issued if a user's role changes.

##### **C.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.



#### **C.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 in most cases might be 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 be 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, 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 shall be part of the signed data.

#### **C.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 shall 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.

#### **C.3.6 Signing time**

The present document provides the capability to include a claimed signing time as an attribute of an electronic signature.

Using this attribute a signer may sign over a time which is the claimed signing time. When an ES with Time-stamp is created (CAAdES-T) then either a trusted time stamp is obtained and added to the ES or a trusted time mark exists in an audit trail. When a verifier accepts a signature, the two times shall be within acceptable limits. In all cases, the claimed signing time cannot be after the time identified by the time-stamp or time-mark.

A further optional attribute is defined in the present document to timestamp the content, to provide proof of the existence of the content, at the time indicated by the time-stamp token.





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 clause 5.11.4).

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

### **C.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 attribute to present the data to the relying party, then a human relying party may misinterpret data with valid signatures.

### **C.3.8 Content cross referencing**

When presenting a signed data is in related to another signed data, it may be important to identify the signed data to which it relates to. The Content-reference and Content-identifier attributes as defined in ESS ([RFC 2634](#) [5]) provide the ability to link a request and reply messages in an exchange between two parties.

## **C.4 Components of validation data**

### **C.4.1 Revocation status information**

A verifier will have to ascertain 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).

NOTE 1: The time of the signature may not be know, so time-stamping or time-marking may be used to provide the time indication of when it was known the signature existed.

NOTE 2: When validating an electronic signature and checking revocation status information a "grace period" is required

which needs to be suitably long enough to allow the involved

authority to process a "last minute" revocation request and for the request to propagate through the revocation system. This grace period is to be added to the time included with the timestamp token or the time mark and thus the revocation status information should be captured after the end of the grace period.

#### **C.4.1.1 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. However, a "grace period" is required to allow CAs time to process revocation requests.

For example, a revocation request may arrive at a CA just before issuing the next CRL and there may not enough time to include the revised revocation status information. This involves checking that the signer certificate serial number is not included in the CRL. The signer, the initial or subsequent verifier may obtain either this CRL. If obtained by the signer, then it shall 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 as an CAvES-C form.

Even if the certificate serial number appears in the CRL with the status "suspended" (i.e. on hold), the signature is not to be deemed as valid since a suspended certificate is not supposed to be used even by its rightful owner.

#### **C.4.1.2 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, still providing a "grace period" suitable enough to allow the involved authority to process a "last minute" revocation request. The signer, the verifier or any other third party may fetch this OCSP response. Since OCSP 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, and

incorporate references to them in the electronic signature itself as an CAdES-C form.

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, same comment as for CRL applies.

#### **C.4.2 Certification path**

A verifier may have to ascertain that the certification path was valid, at the time of the signature, up to a trust point according to the:

- naming constraints;
- certificate policy constraints;
- Signature Policy, when applicable.

Since the time of the signature cannot be known with certainty, an upper limit of it should be used as indicated by either the time stamp or time mark.

In this case 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, when applicable this may be specified as part of the Signature Policy. In addition, it will be necessary to capture the Certificate Authority Revocation Lists (CARLs) to prove that none of the CAs from the chain was revoked at the time of the signature. Again, all this material may be incorporated in the electronic signature (ES X forms). An alternative would be to store it in some storage so that they can be easily retrieved, and incorporate references to it in the electronic signature itself as an CAAdES-C form.

#### **C.4.3 Time-stamping for long life of signatures**

An important property for long standing signatures is that a signature, having been found once to be valid, shall 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 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 the signer's certificate and all the CA certificates used to form a valid certification path 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 were valid at 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 shall

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. Any time stamp or time mark that is taken after the expiration date of any certificate in the certification path has no value in proving the validity of a signature.

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 Basic Electronic Signature, but the essential component of the ES with Time-stamp.

It is required in the present document that if a signer's digital signature value is to be time-stamped, the Time-Stamp Token is issued by a trusted source, known as a Time-stamping Authority.

The present document requires that the signer's digital signature value is time-stamped by a trusted source before the electronic signature can become an ES with Complete validation data. Acceptable TSAs may be specified in a Signature Validation Policy.

This technique is referred to as CAAdES-C in the present document. 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.

#### **C.4.4 Time-stamping for long life of signature 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 at 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 present 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 (CAvES-X Type 1). This format is suitable to be used with an OCSP response and offers the additional advantage to provide an integrity protection over the whole data;
- 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 (CAvES-X Type2). This format is suitable to be used with CRLs, since the time-stamped information may be used for more than one signature (when signers have their certificates issued by the same CA and when signatures can be checked using the same CRLs).

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

#### **C.4.4.1 Time-stamping the ES with complete validation data** (CAvES-X Type 1)

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 CAvES-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 CAAdES-X Type 1 in the present 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 an CAAdES-X Long Type 1 as defined by the present document.

An CAAdES-X Long Type 1 is simply the concatenation of an CAAdES-X Type 1 with a copy of the additional data being referenced.

#### **C.4.4.2 Time-stamping certificates and revocation information references (CAAdES-X Type 2)**

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 exist 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 exist 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.

This technique is referred to as CAAdES-X Type 2 in the present document and requires the following:

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

Thus an CAAdES-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+.



#### **C.4.5 Time-stamping for archive 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 possibility.

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 crypto analysis, or improvements in crypto analytical techniques). Before 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 matters, a single technique, called Archive validation data, covering all the cases is being used in the present document.

Archive validation data consists of the 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 the present document was at least **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.

This technique is referred to as CAvES-A in the present document.



#### **C.4.6 Reference to additional data**

Using CAAdES-X Type 1 or CAAdES-X Type 2 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 (CAAdES-C) is adequate. The actual certificates and CRL information reference in the CAAdES-C can be gathered when needed for arbitration.

If references to additional data are not adequate, then the actual values of all the certificates and revocation information required may be part of the electronic signature. This technique is referred to as CAAdES-X Long Type 1 or CAAdES-X Long Type 2 in the present document.

#### **C.4.7 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 may

be part of a mutually agreed Signature Validation Policy which is part of an agreed signature policy under which digital signature may be used to support the business relationship between the two parties.



#### **C.4.8 TSA key compromise**

TSA servers should be built in such a way that once the private signature key is installed, 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 CAAdES-T and the CAAdES-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 CAAdES-A should be used. For extremely long periods this may be applied repeatedly using new TSA keys.

This technique is referred to as a nested CAAdES-A in the present document.

#### **C.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 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 shall 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 shall be provided.

These forms are described in clause 5.13. 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 D (informative):Data protocols to interoperate with TSPs

## **D.1 Operational protocols**

The following protocols can be used by signers and verifiers to interoperate with Trusted Service Providers during the electronic signature creation and validation.

### **D.1.1 Certificate retrieval**

User certificates, CA certificate and cross-certificates can be retrieved from a repository using the Lightweight Directory Access Protocol as defined in as defined [RFC 2559](#) (see informative references), with the schema defined in [RFC 2587](#) (see informative references).

### **D.1.2 CRL retrieval**

Certificate revocation lists, including authority revocation lists and partial CRL variants, can be retrieved from a repository using the Lightweight Directory Access Protocol as defined in [RFC 2559](#) (see informative references), with the schema defined in [RFC 2587](#) (see informative references).

### **D.1.3 OnLine certificate status**

As an alternative to use of certificate revocation lists the status of certificate can be checked using the OnLine Certificate Status Protocol (OCSP) as defined in [RFC 2560](#) [3].

### **D.1.4 Time-stamping**

The time-stamping service can be accessed using the Time-Stamping Protocol defined in [RFC 3161](#) [7].

## **D.2 Management protocols**

Signers and verifiers can use the following management protocols to manage the use of certificates.

### **D.2.1 Request for certificate revocation**

Request for a certificate to be revoked can be made using the revocation request and response messages defined in [RFC 2510](#) (see informative references).



Annex E (informative): Guidance on naming

**E.1 Allocation of names**

The subject name shall be allocated through a registration scheme administered through a Registration Authority (RA) to ensure uniqueness. This RA may be an independent body or a function carried out by the Certification Authority.

In addition to ensuring uniqueness, the RA shall verify that the name allocated properly identifies the applicant and that authentication checks are carried out to protect against masquerade.

The name allocated by an RA is based on registration information provided by, or relating to, the applicant (e.g. his personal name, date of birth, residence address) and information allocated by the RA. Three variations commonly exist:

- the name is based entirely on registration information which uniquely identifies the applicant (e.g. "Pierre Durand (born on) July 6, 1956");
- the name is based on registration information with the addition of qualifiers added by the registration authority to ensure uniqueness (e.g. "Pierre Durand 12");
- the registration information is kept private by the registration authority and the registration authority allocates a "pseudonym".

**E.2 Providing access to registration information**

Under certain circumstances it may be necessary for information used during registration, but not published in the certificate, to be made available to third parties (e.g. to an arbitrator to resolve a dispute or for law enforcement). This registration information is likely to include personal and sensitive information.

Thus the RA needs to establish a policy for:

- whether the registration information should be disclosed;
- to whom such information should be disclosed;
- under what circumstances such information should be disclosed.

This policy may be different whether the RA is being used only within a company or for public use. The policy will have to take into account national legislation and in particular any data protection and privacy legislation.



Currently, the provision of access to registration is a local matter for the RA. However, if open access is required, standard protocols such as HTTP - [RFC 2068](#) (Internet Web Access Protocol) may be employed with the addition of security mechanisms necessary to meet the data protection requirements (e.g. Transport Layer Security - [RFC 2246](#) with client authentication).

### **[E.3](#) Naming schemes**

#### **[E.3.1](#) Naming schemes for individual citizens**

In some cases the subject name that is contained in a public key certificate may not be meaningful enough. This may happen because of the existence of homonyms or because of the use of pseudonyms. A distinction could be made if more attributes were present. However, adding more attributes to a public key certificate placed in a public repository would be going against the privacy protection requirements.

In any case the Registration Authority will get information at the time of registration but not all that information will be placed in the certificate. In order to achieve a balance between these two opposite requirements the hash values of some additional attributes can be placed in a public key certificate. When the certificate owner provides these additional attributes, then they can be verified. Using biometrics attributes may unambiguously identify a person. Example of biometrics attributes that can be used include: a picture or a manual signature from the certificate owner.

NOTE: Using hash values protects privacy only if the possible inputs are large enough. For example, using the hash of a person's social security number is generally not sufficient since it can easily be reversed.

A picture can be used if the verifier once met the person and later on wants to verify that the certificate that he or she got relates to the person whom was met. In such a case, at the first exchange the picture is sent and the hash contained in the certificate may be used by the verifier to verify that it is the right person. At the next exchange the picture does not need to be sent again.

A manual signature may be used if a signed document has been received beforehand. In such a case, at the first exchange the drawing of the manual signature is sent and the hash contained in the certificate may be used by the verifier to verify that it is the right manual signature. At the next exchange the manual signature does not need to be sent again.





### **E.3.2 Naming schemes for employees of an organization**

The name of an employee within an organization is likely to be some combination of the name of the organization and the identifier of the employee within that organization.

An organization name is usually a registered name, i.e. business or trading name used in day to day business. This name is registered by a Naming Authority, which guarantees that the organization's registered name is unambiguous and cannot be confused with another organization. In order to get more information about a given registered organization name, it is necessary to go back to a publicly available directory maintained by the Naming Authority.

The identifier may be a name or a pseudonym (e.g. a nickname or a employee number). When it is a name, it is supposed to be descriptive enough to unambiguously identify the person. When it is a pseudonym, the certificate does not disclose the identity of the person. However it ensures that the person has been correctly authenticated at the time of registration and therefore may be eligible to some advantages implicitly or explicitly obtained through the possession of the certificate. In either case, however, this can be insufficient because of the existence of homonyms.

Placing more attributes in the certificate may be one solution, for example by giving the organization unit of the person or the name of a city where the office is located. However the more information is placed in the certificate the more problems arise if there is a change in the organization structure or the place of work. So this may not be the best solution. An alternative is to provide more attributes (like the organization unit and the place of work) through access to a directory maintained by the company. It is likely that at the time of registration the Registration Authority got more information than what was placed in the certificate, if such additional information is placed in a repository accessible only to the organization.



Annex F (informative): Example structured contents and MIME

### **F.1 General description**

The signed content may be structured as using MIME (Multipurpose Internet Mail Extensions - [RFC 2045](#) [6]). Whilst the MIME structure was initially developed for Internet e-mail, it has a number of features which make it useful to provide a common structure for encoding a range of electronic documents and other multi-media data (e.g. photographs, video). These features include:

- it provides a means of signalling the type of "object" being carried (e.g. text, image, ZIP file, application data);
- it provides a means of associating a file name with an object;
- it can associate several independent "objects" (e.g. a document and image) to form a multi-part object;
- it can handle data encoded in text or binary and, if necessary, re-encode the binary as text.

When encoding a single object MIME consists of:

- header information, followed by;
- encoded content.

This structure can be extended to support multi-part content.

### **F.2 Header information**

A MIME header includes:

MIME Version information:  
e.g.: MIME-Version: 1.0

Content type information which includes information describing the content sufficient for it to be presented to a user or application process as required. This includes information on the "media type" (e.g. text, image, audio) or whether the data is for passing to a particular type of application. In the case of text the content type includes information on the character set used.

e.g. Content-Type: text/plain; charset="us-ascii"

Content encoding information, which defines how the content is encoded. (See below about encoding supported by MIME).

Other information about the content such as a description, or an

associated file name.

An example MIME header for text object is:

```
Mime-Version: 1.0
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: quoted-printable
```

An example MIME header for a binary file containing a word document is:

```
Content-Type: application/octet-stream
Content-Transfer-Encoding: base64
Content-Description: JCFV201.doc (Microsoft Word Document)
Content-Disposition: filename="JCFV201.doc"
```

### **F.3 Content encoding**

MIME supports a range of mechanisms for encoding the both text and binary data.

Text data can be carried transparently as lines of text data encoded in **7 or 8 bit ASCII characters**. MIME also includes a "quoted-printable" encoding which converts characters other than the basic ASCII into an ASCII sequence.

Binary can either be carried:

- transparently a 8 bit octets; or
- converted to a basic set of characters using a system called Base64.

NOTE: As there are some mail relays which can only handle 7 bit ASCII, Base64 encoding is usually used on the Internet.

### **F.4 Multi-part content**

Several objects (e.g. text and a file attachment) can be associated together using a special "multi-part" content type. This is indicated by the content type "multipart" with an indication of the string to be used indicate a separation between each part.

In addition to a header for the overall multipart content, each part includes its own header information indicating the inner content type and encoding.

An example of a multipart content is:

```
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="----
=_NextPart_000_01BC4599.98004A80"
```

Content-Transfer-Encoding: 7bit

Pinkas, Pope & Ross

[Page 115]

-----=\_NextPart\_000\_01BC4599.98004A80  
Content-Type: text/plain; charset=ISO-8859-1  
Content-Transfer-Encoding: 7bit

Per your request, I've attached our proposal for the Java Card Version [2.0 API and the Java Card FAQ](#).

-----=\_NextPart\_000\_01BC4599.98004A80  
Content-Type: application/octet-stream; name="JCFV201.doc"  
Content-Transfer-Encoding: base64  
Content-Description: JCFV201.doc (Microsoft Word Document)  
Content-Disposition: attachment; filename="JCFV201.doc"

0M8R4KGxGuEAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAAACAAAAAgAAAAA  
AAAAEAAAtAAAAEAAAD+////AAAAAAMAAAAGAAAAA/////////////////////////  
/////////////////AANhAAQAYg==

-----=\_NextPart\_000\_01BC4599.98004A80--

Multipart content can be nested. So a set of associated objects (e.g. HTML text and images) can be handled as a single attachment to another object (e.g. text).

**[F.5 S/MIME](#)**

Previous clauses in this annex have described the use of MIME to encode data. MIME encoded data can be signed (i.e. carried in the eContent of the SignedData structure) thereby signalling the type of information that has been signed.

MIME can also be used to encode the CMS structure containing data after it has been signed so that, for example, this can be carried within an e-mail message. The specific use of MIME to carry CMS (extended as defined in the present document) secured data is called S/MIME. The relationship between the general use of MIME for encoding content, CMS and S/MIME is illustrated in figure F.1.

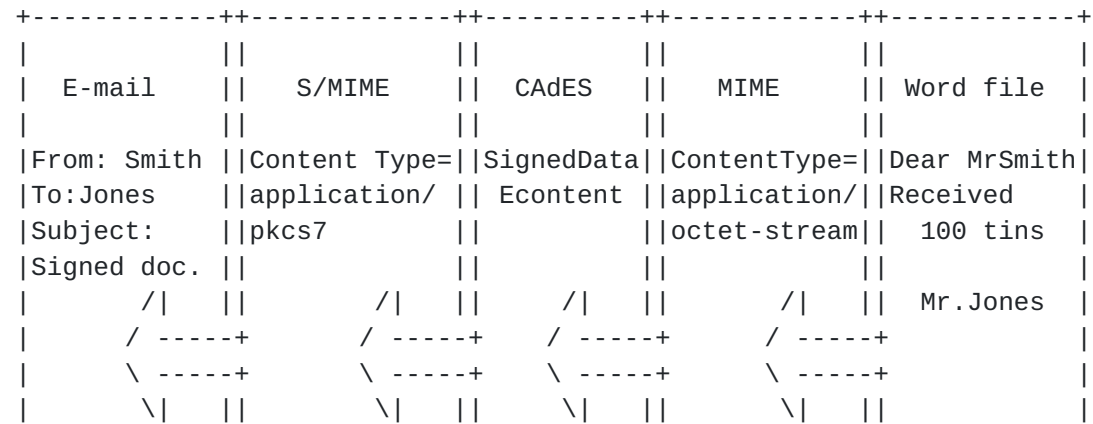




Figure F.1



S/MIME carries electronic signatures as either:

- an "application/pkcs7-mime" object with the CMS carried as binary attachment (PKCS7 is the name of the early version of CMS).

An example of signed data encoded using this approach is:

Content-Type: application/pkcs7-mime; smime-type=signed-data;

Content-Transfer-Encoding: base64

Content-Disposition: attachment; filename=smime.p7m

```
567GhIGfHfYT6ghyHhHUujpfyF4f8HHGTrfvhJhjH776tbB9HG4VQbnj7
77n8HHGT9HG4VQpfyF467GhIGfHfYT6rfvbnj756tbBghyHhHUujhJhjH
HUujhJh4VQpfyF467GhIGfHfYGT6rfvbnjT6jH7756tbB9H7n8HHGghyHh
6YT64V0GhIGfHfQbnj75
```

This approach is similar to handling signed data as any other binary file attachment. Thus, this encoding can be used where signed data passes through gateways to other e-mail systems (e.g. those based on other e-mail systems).

A "multipart/signed" object with the signed data and the signature encoded as separate MIME objects.

An example of signed data encoded this approach is:

Content-Type: multipart/signed;

protocol="application/pkcs7-signature";

micalg=sha1; boundary=boundary42

--boundary42

Content-Type: text/plain

This is a clear-signed message.

--boundary42

Content-Type: application/pkcs7-signature; name=smime.p7s

Content-Transfer-Encoding: base64

Content-Disposition: attachment; filename=smime.p7s

```
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756
```

--boundary42--



With this second approach MIME the signed data passes through the CMS process and is carried as part of the S/MIME structure as illustrated in figure F.2. The CMS structure just holds the electronic signature.



Figure F.2

The second approach (multipart/signed) has the advantage that the signed data can be decoded by any MIME compatible e-mail system even if it does not recognize CMS encoded electronic signatures. However, this form cannot be used with other e-mail systems.



Annex G (informative): Relationship to the European Directive and EESSI

**G.1 Introduction**

This annex provides an indication of the relationship between electronic signatures created under the present document and requirements under the European Parliament and Council Directive on a Community framework for electronic signatures.

NOTE: Legal advice should be sought on the specific national legislation regarding use of electronic signatures.

The present document is one of a set of standards being defined under the "European Electronic Signature Standardization Initiative" (EESSI) for electronic signature products and solutions compliant with the European Directive for electronic signatures.

**G.2 Electronic signatures and the directive**

This directive defines electronic signatures as:

- "data in electronic form which are attached to or logically associated with other electronic data and which serve as a method of authentication".

The directive states that an electronic signature should not be denied "legal effectiveness and admissibility as evidence in legal proceedings" solely on the grounds that it is in electronic form.

The directive identifies an electronic signature as having equivalence to a hand-written signature if it meets specific criteria:

- it is an "advanced electronic signature" with the following properties:
  - a) it is uniquely linked to the signatory;
  - b) it is capable of identifying the signatory;
  - c) it is created using means that the signatory can maintain under his sole control; and
  - d) it is linked to the data to which it relates in such a manner that any subsequent change of the data is detectable.
- it is based on a certificate which meets detailed criteria given in annex I to the directive and is issued by a "certification-service-provider" which meets requirements given in annex II to the directive. Such a certificate is referred to as a "qualified

certificate";

- it is created by a "device" which detailed criteria given in annex III to the directive. Such a device is referred to a "secure-signature-creation device";

This form of electronic signature is referred to as a "qualified electronic signature" in EESSI (see below).

### **G.3 ETSI electronic signature formats and the directive**

An electronic signature created in accordance with the present document is:

- a) considered to be an "electronic signature" under the terms of the Directive;
- b) considered to be an "advanced electronic signature" under the terms of the Directive;
- c) considered to be a "Qualified Electronic Signature" provided the additional requirements in annex I, II and III of the Directive are met. The requirements in annex I, II and III of the Directive are outside the scope of the present document, and are subject to further standardization.

### **G.4 EESSI standards and classes of electronic signature**

#### **G.4.1 Structure of EESSI standardization**

EESSI looks at standards in several areas. See the ETSI ESI and CEN web sites for the latest list of standards and their versions

- use of X.509 public key certificates as qualified certificates;
- security Management and Certificate Policy for CSPs Issuing Qualified Certificates;
- security requirements for trustworthy systems used by CSPs Issuing Qualified Certificates;
- security requirements for Secure Signature Creation Devices;
- security requirements for Signature Creation Systems;
- procedures for Electronic Signature Verification;
- electronic signature syntax and encoding formats;
- protocol to interoperate with a Time Stamping Authority;
- Policy requirements for Time-Stamping Authorities;

- XML electronic signature formats.



Each of these standards addresses a range of requirements including the requirements of Qualified Electronic Signatures as specified in article [5.1 of the Directive](#). However, some of them also address general requirements of electronic signatures for business and electronic commerce which all fall into the category of article 5.2 of the Directive. Such variation in the requirements may be identified either as different levels or different options.

#### [G.4.2](#) **Classes of electronic signatures**

Since some of these standards address a range of requirements, it may be useful to identify a set of standards to address a specific business need. Such a set of standards and their uses defines a class of electronic signature. The first class already identified is the qualified electronic signature, fulfilling the requirements of article [5.1 of the Directive](#).

A limited number of "classes of electronic signatures" and corresponding profiles could be defined by EESSI, in close co-operation with actors on the market (business, users, suppliers). Need for such standards is envisaged, in addition to those for qualified electronic signatures, in areas such as:

- different classes of electronic signatures with long term validity;
- electronic signatures for business transactions with limited value.

#### [G.4.3](#) **EESSI classes and the ETSI electronic signature format**

The electronic signature format defined in the present document is applicable to the EESSI area "electronic signature and encoding formats".

An electronic signature produced by a signer (see clause 5 and conformance clause 10.1) is applicable to the proposed class of electronic signature: "qualified electronic signatures fulfilling article 5.1".

With the addition of validation data by the verifier (see clause 6 and conformance clause 10.2) this would become applicable electronic signatures adding long-term validity attributes to the qualified electronic signature.

Annex H (informative): APIs for the generation and verification of electronic signatures tokens

While the present document describes the data format of an electronic

signature, the question is whether there exists APIs (Application Programming Interfaces) able to manipulate these structures. At least two such APIs have been defined. One set by the IETF and another set by the OMG (Object Management Group).

## **H.1 Data framing**

In order to be able to use either of these APIs, it will be necessary to frame the previously defined electronic signature data structures using a mechanism-independent token format. Clause 3.1 of [RFC 2743](#) (see informative references) describes that framing incorporating an identifier of the mechanism type to be used and enabling tokens to be interpreted unambiguously.

In order to be processable by these APIs, all electronic signature data formats that are defined in the present document shall be framed following that description.

The encoding format for the token tag is derived from ASN.1 and DER, but its concrete representation is defined directly in terms of octets rather than at the ASN.1 level in order to facilitate interoperable implementation without use of general ASN.1 processing code. The token tag consists of the following elements, in order:

- 1) 0x60 -- Tag for [RFC 2743](#) SEQUENCE; indicates that constructed form, definite length encoding follows.
- 2) Token length octets, specifying length of subsequent data (i.e. the summed lengths of elements 3 to 5 in this list, and of the mechanism-defined token object following the tag). This element comprises a variable number of octets:
  - a) If the indicated value is less than 128, it shall be represented in a single octet with bit 8 (high order) set to "0" and the remaining bits representing the value.
  - b) If the indicated value is 128 or more, it shall be represented in two or more octets, with bit 8 of the first octet set to "1" and the remaining bits of the first octet specifying the number of additional octets. The subsequent octets carry the value, 8 bits per octet, most significant digit first. The minimum number of octets shall be used to encode the length (i.e. no octets representing leading zeros shall be included within the length encoding).
- 3) 0x06 -- Tag for OBJECT IDENTIFIER.
- 4) Object identifier length -- length (number of octets) of the encoded object identifier contained in element 5, encoded per rules as described in 2a) and 2b) above.
- 5) object identifier octets -- variable number of octets, encoded per ASN.1 BER rules:



- The first octet contains the sum of two values:

- (1) the top-level object identifier component, multiplied by 40 (decimal); and

- (2) the second-level object identifier component.

This special case is the only point within an object identifier encoding where a single octet represents contents of more than one component.

- Subsequent octets, if required, encode successively-lower components in the represented object identifier. A component's encoding may span multiple octets, encoding 7 bits per octet (most significant bits first) and with bit 8 set to "1" on all but the final octet in the component's encoding. The minimum number of octets shall be used to encode each component (i.e. no octets representing leading zeros shall be included within a component's encoding).

NOTE: In many implementations, elements 3 to 5 may be stored and referenced as a contiguous string constant.

The token tag is immediately followed by a mechanism-defined token object. Note that no independent size specifier intervenes following the object identifier value to indicate the size of the mechanism-defined token object.

Tokens conforming to the present document shall have the following OID in order to be processable by IDUP-APIs:

```
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) }
```

## **H.2 IDUP-GSS-APIs defined by the IETF**

The IETF CAT WG has produced in December 1998 an RFC ([RFC 2479](#) - see informative references) under the name of IDUP-GSS-API (Independent Data Unit Protection) able to handle the electronic signature data format defined in the present document.

The IDUP-GSS-API includes support for non-repudiation services.

It supports evidence generation, where "evidence" is information that either by itself, or when used in conjunction with other information, is used to establish proof about an event or action, as well a evidence verification.



IDUP supports various types of evidences. All the types defined in IDUP are supported in the present document through the commitment type parameter.

Clause 2.3.3 of IDUP describes the specific calls needed to handle evidences ("EV" calls). The "EV" group of calls provides a simple, high-level interface to underlying IDUP mechanisms when application developers need to deal only with evidences but not with encryption or integrity services.

All generations and verification are performed according to the content of a NR policy that is referenced in the context.

Get\_token\_details is used to return to an application the attributes that correspond to a given input token. Since IDUP-GSS- API tokens are meant to be opaque to the calling application, this function allows the application to determine information about the token without having to violate the opaqueness intention of IDUP. Of primary importance is the mechanism type, which the application can then use as input to the IDUP\_Establish\_Env() call in order to establish the correct environment in which to have the token processed.

Generate\_token generates a non-repudiation token using the current environment.

Verify\_evidence verifies the evidence token using the current environment. This operation returns a major\_status code which can be used to determine whether the evidence contained in a token is complete (i.e. can be successfully verified (perhaps years) later). If a token's evidence is not complete, the token can be passed to another API: form\_complete\_pidu to complete it. This happens when a status "conditionally valid" is returned. That status corresponds to the status "validation incomplete" of the present document.

Form\_complete\_PIDU is used primarily when the evidence token itself does not contain all the data required for its verification and it is anticipated that some of the data not stored in the token may become unavailable during the interval between generation of the evidence token and verification unless it is stored in the token. The Form\_Complete\_PIDU operation gathers the missing information and includes it in the token so that verification can be guaranteed to be possible at any future time.

### **H.3 CORBA security interfaces defined by the OMG**

Non-repudiation interfaces have been defined in "CORBA Security", a document produced by the OMG (Object Management Group). These interfaces are described in IDL (Interface Definition Language) and are optional.





The handling of "tokens" supporting non-repudiation is done through the following interfaces:

- set\_NR\_features specifies the features to apply to future evidence generation and verification operations;
- get\_NR\_features returns the features which will be applied to future evidence generation and verification operations;
- generate\_token generates a Non-repudiation token using the current Non-repudiation features;
- verify\_evidence verifies the evidence token using the current Non-repudiation features;
- get\_tokens\_details returns information about an input Non-repudiation token. The information returned depends upon the type of token;
- form\_complete\_evidence is used when the evidence token itself does not contain all the data required for its verification, and it is anticipated that some of the data not stored in the token may become unavailable during the interval between generation of the evidence token and verification unless it is stored in the token. The form\_complete\_evidence operation gathers the missing information and includes it in the token so that verification can be guaranteed to be possible at any future time.

NOTE: The similarity between the two sets of APIs is noticeable.



Annex I (informative):Cryptographic algorithms

[RFC 3370](#) [[10](#)] describes the conventions for using several cryptographic algorithms with the Cryptographic Message Syntax (CMS). Only the hashing and signing algorithms are appropriate for use with the present document.

Since the publication of [RFC 3370](#) [[10](#)], MD5 has been broken. This algorithm is no more considered as appropriate and has been deleted from the list of algorithms.

## **[I.1](#) Digest algorithms**

### **[I.1.1](#) SHA-1**

The SHA-1 digest algorithm is defined in FIPS Pub 180-1. The algorithm identifier for SHA-1 is:

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

The AlgorithmIdentifier parameters field is optional. If present, the parameters field shall contain an ASN.1 NULL. Implementations should accept SHA-1 AlgorithmIdentifiers with absent parameters as well as NULL parameters. Implementations should generate SHA-1 AlgorithmIdentifiers with NULL parameters.

### **[I.1.2](#) General**

The following is a selection of work that has been done in the area of digest algorithms or, as they are often called, hash functions:

- ISO/IEC 10118-1 (1994): "Information technology - Security techniques - Hash-functions - Part 1: General". ISO/IEC 10118-1 contains definitions and describes basic concepts.
- ISO/IEC 10118-2 (1994): "Information technology - Security techniques - Hash-functions - Part 2: Hash-functions using an n-bit block cipher algorithm". ISO/IEC 10118-2 specifies two ways to construct a hash-function from a block cipher.
- ISO/IEC 10118-3 (1997): "Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions". ISO/IEC 10118-3 specifies the following dedicated hash-functions:
  - SHA-1 (FIPS 180-1);
  - RIPEMD-128;
  - RIPEMD-160.



- ISO/IEC 10118-4 (1998): "Information technology - Security techniques - Hash-functions - Part 4: Hash-functions using modular arithmetic".
- [RFC 1320](#) (PS 1992): "The MD4 Message-Digest Algorithm". [RFC 1320](#) specifies the hash-function MD4. Today, MD4 is considered out-dated.
- [RFC 1321](#) (I 1992): "The MD5 Message-Digest Algorithm". [RFC 1321](#) (informational) specifies the hash-unction MD5.
- FIPS Publication 180-1 (1995): "Secure Hash Standard". FIPS 180-1 specifies the Secure Hash Algorithm (SHA), dedicated hash-function developed for use with the DSA. The original SHA published in 1993 was slightly revised in 1995 and renamed SHA-1.
- ANSI X9.30-2 (1997): "Public Key Cryptography for the Financial Services Industry - Part 2: The Secure Hash Algorithm (SHA-1)". X9.30-2 specifies the ANSI-Version of SHA-1.
- ANSI X9.31-2 (1996): "Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry - Part 2: Hash Algorithms". X9.31-2 specifies hash algorithms.

## **[I.2](#) Digital signature algorithms**

### **[I.2.1](#) DSA**

The DSA signature algorithm is defined in FIPS Pub 186. DSA is always used with the SHA-1 message digest algorithm. The algorithm identifier for DSA is:

```
id-dsa-with-sha1 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840)
x9-57 (10040) x9cm(4) 3 }
```

The AlgorithmIdentifier parameters field shall not be present.

### **[I.2.2](#) RSA**

The RSA signature algorithm is defined in [RFC 2437](#) (see informative references). [RFC 3370](#) [[10](#)] specifies the use of the RSA signature algorithm with the SHA-1 algorithm. The algorithm identifier for RSA with SHA-1 is:

```
Sha1WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-1(1) 5 }
```

NOTE: [RFC 3370](#) [[10](#)] recommends that MD5 is not used for new implementations.



### **I.2.3 General**

The following is a selection of work that has been done in the area of digital signature mechanisms:

- FIPS Publication 186 (1994): "Digital Signature Standard". NIST's Digital Signature Algorithm (DSA) is a variant of ElGamal's Discrete Logarithm based digital signature mechanism. The DSA requires a 160-bit hash-function and mandates SHA-1.
- IEEE P1363 (2000): "Standard Specifications for Public-Key Cryptography". IEEE P1363 contains mechanisms for digital signatures, key establishment, and encipherment based on three families of public-key schemes:
- "Conventional" Discrete Logarithm (DL) based techniques, i.e. Diffie-Hellman (DH) key agreement, Menezes-Qu-Vanstone (MQV) key agreement, the Digital Signature Algorithm (DSA), and Nyberg-Rueppel (NR) digital signatures;
- Elliptic Curve (EC) based variants of the DL-mechanisms specified above, i.e. EC-DH, EC-MQV, EC-DSA, and EC-NR. For elliptic curves, implementation options include mod p and characteristic 2 with polynomial or normal basis representation;
- Integer Factoring (IF) based techniques including RSA encryption, RSA digital signatures, and RSA-based key transport.
- ISO/IEC 9796 (1991): "Information technology - Security techniques - Digital signature scheme giving message recovery". ISO/IEC 9796 specifies a digital signature mechanism based on the RSA public-key technique and a specifically designed redundancy function.
- ISO/IEC 9796-2 (1997): "Information technology - Security techniques - Digital signature schemes giving message recovery - Part 2: Mechanisms using a hash-function". ISO/IEC 9796-2 specifies digital signature mechanisms with partial message recovery that are also based on the RSA technique but make use of a hash-function.
- ISO/IEC 9796-4 (1998): "Digital signature schemes giving message recovery - Part 4: Discrete logarithm based mechanisms". ISO/IEC 9796-4 specifies digital signature mechanisms with partial message recovery that are based on Discrete Logarithm techniques. The document includes the Nyberg-Rueppel scheme.

- ISO/IEC 14888-1: "Digital signatures with appendix - Part 1: General". ISO/IEC 14888-1 contains definitions and describes the basic concepts of digital signatures with appendix.



- ISO/IEC 14888-2: "Digital signatures with appendix - Part 2: Identity-based mechanisms". ISO/IEC 14888-2 specifies digital signature schemes with appendix that make use of identity-based keying material. The document includes the zero-knowledge techniques of Fiat-Shamir and Guillou-Quisquater.
- ISO/IEC 14888-3: "Digital signatures with appendix - Part 3: Certificate-based mechanisms". ISO/IEC 14888-3 specifies digital signature schemes with appendix that make use of certificate-based keying material. The document includes five schemes:
  - DSA;
  - EC-DSA, an elliptic curve based analog of NIST's Digital Signature Algorithm;
  - Pointcheval-Vaudeney signatures;
  - RSA signatures;
  - ESIGN.
- ISO/IEC 15946-2 (2002) : "Cryptographic techniques based on elliptic curves - Part 2: Digital signatures".
- ISO/IEC 15946-3 (2002) specifies digital signature schemes with appendix using elliptic curves.
- The document includes two schemes:
  - EC-DSA, an elliptic curve based analog of NIST's Digital Signature Algorithm;
  - EC-AMV, an elliptic curve based analog of the Agnew-Muller-Vanstone signature algorithm.
- ANSI X9.31-1 (1997): "Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry - Part 1: The RSA Signature Algorithm". ANSI X9.31-1 specifies a digital signature mechanism with appendix using the RSA public-key technique.
- ANSI X9.30-1 (1997): "Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry - Part 1: The Digital Signature Algorithm (DSA)". ANSI X9.30-1 specifies the DSA, NIST's Digital Signature Algorithm.
- ANSI X9.62 (1998): "Public Key Cryptography for the Financial Services Industry - The Elliptic Curve Digital Signature Algorithm (ECDSA)". ANSI X9.62 specifies the Elliptic Curve Digital Signature Algorithm, an analog of NIST's Digital Signature Algorithm (DSA) using elliptic curves. The appendices provide tutorial information on the underlying mathematics for elliptic curve cryptography and many examples.



Annex J (informative): Changes from the previous version

The title of the document has changed to be aligned with the title of XAdES, the vocabulary used within the present document has been aligned with the vocabulary used in XAdES,

In the previous version of TS 101 733 (i.e. version 1.5.1) sigPolicyHash was mandatory. Implementations requiring to be backward compatible with version 1.5.1 and previous versions of the current document MUST include SigPolicyHash.

The OIDs from the ASN.1 modules have changed for the following reasons:

- the OIDs of the ASN.1 modules of [RFC 2560](#) and [RFC 3161](#) have been included.
- since [RFC 2459](#) and [RFC 3369](#) has been obsoleted by [RFC 3280](#) and [RFC 3852](#) respectively, there was the need to refer to the OIDs of the ASN.1 modules of [RFC 3280](#) and [RFC 3852](#), instead of the OIDs of the ASN.1 modules of [RFC 2459](#) and [RFC 3369](#).
- the other change is related to the field sigPolicyHash from SignaturePolicyId (see clause 5.8.1). That field was mandatory and is now optional.



## Full Copyright Statement

Copyright (C) The Internet Society (2005).

The contents of this Informational RFC amounts to a transposition of the ETSI TS 101 733 V.1.6.3 and is technically equivalent to it. The ETSI TS is under the ETSI Copyright (C). Individual copies of this ETSI deliverable can be downloaded from <http://www.etsi.org>.

This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

## Disclaimer

This document and the information contained herein are provided on An "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at [ietf-ipr@ietf.org](mailto:ietf-ipr@ietf.org).

ETSI takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed

to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights.

Information on the ETSI Intellectual Property Rights Policy may be obtained from <<http://www.etsi.org/legal/home.htm>>. The document is an extract from Annex 6 of the ETSI Rules of Procedure that are available at : <<http://portal.etsi.org/directives/home.asp>>.

The ETSI IPR database <http://webapp.etsi.org/IPR/home.asp> contains IPRs, particularly patents and patent applications, which have been notified to ETSI as being essential, or potentially essential, to ETSI standards. Unless otherwise specified, all IPRs contained herein have been notified to ETSI, with an undertaking from the owner to grant licenses according to the terms and conditions of Article 6.1 of Annex 6 of the ETSI Rules of the ETSI IPR Policy.

The ETSI IPR database provides data that is based on the information received. ETSI has not checked the validity of the information, nor the relevance of the identified patents/patent applications to the ETSI Standards and cannot confirm, or deny, that the patents/patent applications are, in fact, essential, or potentially essential. No investigation, or IPR searches, have been carried out by ETSI and therefore no guarantee can be given concerning the existence of other IPRs which are, or may become, essential.

Potential Licensees should use the information in this database at their discretion and should contact the patent holder.

