

Network Working Group  
Request for Comments: 5126  
Obsoletes: 3126  
Category: Informational

D. Pinkas  
Bull SAS  
N. Pope  
Thales eSecurity  
J. Ross  
Security and Standards  
February 2008

## CMS Advanced Electronic Signatures (CAeS)

### Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

### 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 3852 and RFC 2634, where, when appropriate, additional signed and unsigned attributes have been defined.

The contents of this Informational RFC amount to a transposition of the ETSI Technical Specification (TS) 101 733 V.1.7.4 (CMS Advanced Electronic Signatures -- CAeS) and is technically equivalent to it.

The technical contents of this specification are maintained by ETSI. The ETSI TS and further updates are available free of charge at:  
<http://www.etsi.org/Website/Standards/StandardsDownload.aspx>

## Table of Contents

1. Introduction .....	6
2. Scope .....	6
3. Definitions and Abbreviations .....	8
3.1. Definitions .....	8
3.2. Abbreviations .....	11
4. Overview .....	12
4.1. Major Parties .....	13
4.2. Signature Policies .....	14
4.3. Electronic Signature Formats .....	15
4.3.1. CAAdES Basic Electronic Signature (CAAdES-BES) .....	15
4.3.2. CAAdES Explicit Policy-based Electronic Signatures (CAAdES-EPES) .....	18
4.4. Electronic Signature Formats with Validation Data .....	19
4.4.1. Electronic Signature with Time (CAAdES-T) .....	20
4.4.2. ES with Complete Validation Data References (CAAdES-C) .....	21
4.4.3. Extended Electronic Signature Formats .....	23
4.4.3.1. EXTENDED Long Electronic Signature (CAAdES-X Long) .....	24
4.4.3.2. EXTENDED Electronic Signature with Time Type 1 .....	25
4.4.3.3. EXTENDED Electronic Signature with Time Type 2 .....	26
4.4.3.4. EXTENDED Long Electronic Signature with Time (CAAdES-X Long) .....	27
4.4.4. Archival Electronic Signature (CAAdES-A) .....	27
4.5. Arbitration .....	28
4.6. Validation Process .....	29
5. Electronic Signature Attributes .....	30
5.1. General Syntax .....	30
5.2. Data Content Type .....	30
5.3. Signed-data Content Type .....	30
5.4. SignedData Type .....	31
5.5. EncapsulatedContentInfo Type .....	31
5.6. SignerInfo Type .....	31
5.6.1. Message Digest Calculation Process .....	32
5.6.2. Message Signature Generation Process .....	32
5.6.3. Message Signature Verification Process .....	32
5.7. Basic ES Mandatory Present Attributes .....	32
5.7.1. content-type .....	32
5.7.2. Message Digest .....	33
5.7.3. Signing Certificate Reference Attributes .....	33
5.7.3.1. ESS signing-certificate Attribute Definition .....	34
5.7.3.2. ESS signing-certificate-v2 Attribute Definition .....	34

5.7.3.3. Other signing-certificate Attribute Definition .....	35
5.8. Additional Mandatory Attributes for Explicit Policy-based Electronic Signatures .....	36
5.8.1. signature-policy-identifier .....	36
5.9. CMS Imported Optional Attributes .....	38
5.9.1. signing-time .....	38
5.9.2. countersignature .....	39
5.10. ESS-Imported Optional Attributes .....	39
5.10.1. content-reference Attribute .....	39
5.10.2. content-identifier Attribute .....	39
5.10.3. content-hints Attribute .....	40
5.11. Additional Optional Attributes Defined in the Present Document .....	40
5.11.1. commitment-type-indication Attribute .....	41
5.11.2. signer-location Attribute .....	43
5.11.3. signer-attributes Attribute .....	43
5.11.4. content-time-stamp Attribute .....	44
5.12. Support for Multiple Signatures .....	44
5.12.1. Independent Signatures .....	44
5.12.2. Embedded Signatures .....	45
6. Additional Electronic Signature Validation Attributes .....	45
6.1. signature time-stamp Attribute (CADES-T) .....	47
6.1.1. signature-time-stamp Attribute Definition .....	47
6.2. Complete Validation Data References (CADES-C) .....	48
6.2.1. complete-certificate-references Attribute Definition .....	48
6.2.2. complete-revocation-references Attribute Definition .....	49
6.2.3. attribute-certificate-references Attribute Definition .....	51
6.2.4. attribute-revocation-references Attribute Definition .....	52
6.3. Extended Validation Data (CADES-X) .....	52
6.3.1. Time-Stamped Validation Data (CADES-X Type 1 or Type 2) .....	53
6.3.2. Long Validation Data (CADES-X Long, CADES-X Long Type 1 or 2) .....	53
6.3.3. certificate-values Attribute Definition .....	54
6.3.4. revocation-values Attribute Definition .....	54
6.3.5. CADES-C-time-stamp Attribute Definition .....	56
6.3.6. time-stamped-certs-crls-references Attribute Definition .....	57
6.4. Archive Validation Data .....	58
6.4.1. archive-time-stamp Attribute Definition .....	58
7. Other Standard Data Structures .....	60
7.1. Public Key Certificate Format .....	60
7.2. Certificate Revocation List Format .....	60

7.3.	OCSF Response Format .....	60
7.4.	Time-Stamp Token Format .....	60
7.5.	Name and Attribute Formats .....	60
7.6.	AttributeCertificate .....	61
8.	Conformance Requirements .....	61
8.1.	CADES-Basic Electronic Signature (CADES-BES) .....	62
8.2.	CADES-Explicit Policy-based Electronic Signature .....	63
8.3.	Verification Using Time-Stamping .....	63
8.4.	Verification Using Secure Records .....	63
9.	References .....	64
9.1.	Normative References .....	64
9.2.	Informative References .....	65
Annex A	(normative): ASN.1 Definitions .....	69
A.1.	Signature Format Definitions Using X.208 ASN.1 Syntax .....	69
A.2.	Signature Format Definitions Using X.680 ASN.1 Syntax .....	77
Annex B	(informative): Extended Forms of Electronic Signatures ...	86
B.1.	Extended Forms of Validation Data .....	86
B.1.1.	CADES-X Long .....	87
B.1.2.	CADES-X Type 1 .....	88
B.1.3.	CADES-X Type 2 .....	90
B.1.4.	CADES-X Long Type 1 and CADES-X Long Type 2 ...	91
B.2.	Time-Stamp Extensions .....	93
B.3.	Archive Validation Data (CADES-A) .....	94
B.4.	Example Validation Sequence .....	97
B.5.	Additional Optional Features .....	102
Annex C	(informative): General Description .....	103
C.1.	The Signature Policy .....	103
C.2.	Signed Information .....	104
C.3.	Components of an Electronic Signature .....	104
C.3.1.	Reference to the Signature Policy .....	104
C.3.2.	Commitment Type Indication .....	105
C.3.3.	Certificate Identifier from the Signer .....	106
C.3.4.	Role Attributes .....	106
C.3.4.1.	Claimed Role .....	107
C.3.4.2.	Certified Role .....	107
C.3.5.	Signer Location .....	108
C.3.6.	Signing Time .....	108
C.3.7.	Content Format .....	108
C.3.8.	content-hints .....	109
C.3.9.	Content Cross-Referencing .....	109
C.4.	Components of Validation Data .....	109
C.4.1.	Revocation Status Information .....	109
C.4.1.1.	CRL Information .....	110
C.4.1.2.	OCSF Information .....	110
C.4.2.	Certification Path .....	111
C.4.3.	Time-stamping for Long Life of Signatures ...	111

C.4.4.	Time-stamping for Long Life of Signature before CA key Compromises .....	113
C.4.4.1.	Time-stamping the ES with Complete Validation Data .....	113
C.4.4.2.	Time-Stamping Certificates and Revocation Information References ..	114
C.4.5.	Time-stamping for Archive of Signature .....	115
C.4.6.	Reference to Additional Data .....	116
C.4.7.	Time-Stamping for Mutual Recognition .....	116
C.4.8.	TSA Key Compromise .....	117
C.5.	Multiple Signatures .....	118
Annex D (informative):	Data Protocols to Interoperate with TSPs ..	118
D.1.	Operational Protocols .....	118
D.1.1.	Certificate Retrieval .....	118
D.1.2.	CRL Retrieval .....	118
D.1.3.	Online Certificate Status .....	119
D.1.4.	Time-Stamping .....	119
D.2.	Management Protocols .....	119
D.2.1.	Request for Certificate Revocation .....	119
Annex E (informative):	Security Considerations .....	119
E.1.	Protection of Private Key .....	119
E.2.	Choice of Algorithms .....	119
Annex F (informative):	Example Structured Contents and MIME .....	120
F.1.	General Description .....	120
F.1.1.	Header Information .....	120
F.1.2.	Content Encoding .....	121
F.1.3.	Multi-Part Content .....	121
F.2.	S/MIME .....	122
F.2.1.	Using application/pkcs7-mime .....	123
F.2.2.	Using application/pkcs7-signature .....	124
Annex G (informative):	Relationship to the European Directive and EESSI .....	125
G.1.	Introduction .....	125
G.2.	Electronic Signatures and the Directive .....	126
G.3.	ETSI Electronic Signature Formats and the Directive ..	127
G.4.	EESSI Standards and Classes of Electronic Signature ..	127
G.4.1.	Structure of EESSI Standardization .....	127
G.4.2.	Classes of Electronic Signatures .....	128
G.4.3.	Electronic Signature Classes and the ETSI Electronic Signature Format .....	128
Annex H (informative):	APIs for the Generation and Verification of Electronic Signatures Tokens .....	129
H.1.	Data Framing .....	129
H.2.	IDUP-GSS-APIs Defined by the IETF .....	131
H.3.	CORBA Security Interfaces Defined by the OMG .....	132
Annex I (informative):	Cryptographic Algorithms .....	133
I.1.	Digest Algorithms .....	133
I.1.1.	SHA-1 .....	133

- I.1.2. General .....133
- I.2. Digital Signature Algorithms .....134
  - I.2.1. DSA .....134
  - I.2.2. RSA .....135
  - I.2.3. General .....135
- Annex J (informative): Guidance on Naming .....137
  - J.1. Allocation of Names .....137
  - J.2. Providing Access to Registration Information .....138
  - J.3. Naming Schemes .....138
    - J.3.1. Naming Schemes for Individual Citizens .....138
    - J.3.2. Naming Schemes for Employees of an Organization .....139

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 [ISO10181-5]) 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, Global System for Mobile Communication Subscriber Identity Module (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.

2. Scope

The scope of the present document covers electronic signature formats only. The aspects of Electronic Signature Policies are defined in RFC 3125 [RFC3125] and ETSI TR 102 272 [TR102272].

The present document defines a number of electronic signature formats, including electronic signatures 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, as 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 it 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. This document 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. These include:

- 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 Certificate Revocation List (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) [14]. ASN.1 is encoded using X.690 [16].

These formats are based on CMS (Cryptographic Message Syntax) defined in RFC 3852 [4]. These electronic signatures are thus called CAdES, for "CMS Advanced Electronic Signatures".

Another document, TS 101 903 [TS101903], describes formats for XML advanced electronic signatures (XAdES) built on XMLDSIG as specified in [XMLDSIG].

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

Informative annexes include:

- illustrations of extended forms of Electronic Signature formats that protect against various vulnerabilities and examples of validation processes (Annex B);
- descriptions and explanations of some of the concepts used in the present document, giving a rationale for normative parts of the present document (Annex C);
- information on protocols to interoperate with Trusted Service Providers (Annex D);
- guidance on naming (Annex E);
- an example structured content and MIME (Annex F);
- the relationship between the present document and the directive on electronic signature and associated standardization initiatives (Annex G);
- APIs to support the generation and verification of electronic signatures (Annex H);
- cryptographic algorithms that may be used (Annex I); and
- naming schemes (see Annex J).

### 3. Definitions and Abbreviations

#### 3.1. Definitions

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

**Arbitrator:** an 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): an authority that assigns privileges by issuing attribute certificates.

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

Attribute Authority Revocation List (AARL): a 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): a 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): a 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): an 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): a 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 [ISO7498-2].

Electronic Signature: data in electronic form that is attached to or logically associated with other electronic data and that serves 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 [EUDirective].

Extended 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 that shall be used to validate it is explicitly specified.

**Grace Period:** a time period that 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 that issued it.

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

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

**Signature Policy:** a 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:** an entity that defines and issues a signature policy.

**Signature Validation Policy:** part of the signature policy that specifies the technical requirements on the signer in creating a signature and verifier when validating a signature.

**Signer:** an entity that creates an electronic signature.

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

NOTE: Subsequent verification 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:** a 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:** a 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):** a 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):** a set of hardware and software that is managed as a unit and has a single time-stamp token signing key active at a time.

**Trusted Service Provider (TSP):** an 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 that the signature is valid.

**Valid Electronic Signature:** an electronic signature that passes validation.

**Verifier:** an entity that verifies evidence.

NOTE 1: See ISO/IEC 13888-1 [ISO13888-1].

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
CAdES	CMS Advanced Electronic Signature
CAdES-A	CAdES with Archive validation data

CADES-BES	CADES Basic Electronic Signature
CADES-C	CADES with Complete validation data
CADES-EPES	CADES Explicit Policy Electronic Signature
CADES-T	CADES with Time
CADES-X	CADES with eXtended validation data
CADES-X Long	CADES with eXtended Long validation data
CARL	Certification Authority Revocation List
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
CWA	CEN (European Committee for Standardization) 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	Public Key Infrastructure using X.509 (IETF Working Group)
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	Extensible Markup Language
XMLDSIG	XML Digital Signature

#### 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 section:

- provides an introduction to the major parties involved (Section 4.1),

- introduces the concept of signature policies (Section 4.2),
- provides an overview of the various ES formats (Section 4.3),
- introduces the concept of validation data, and provides an overview of formats that incorporate validation data (Section 4.4), and
- presents relevant considerations on arbitration (Section 4.5) and for the validation process (Section 4.6).

The formal specifications of the attributes are specified in Sections 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); and
- 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, and OCSP responses. The following TSPs are used to support the functions defined in the present document:

- Certification Authorities;
- Registration Authorities;
- CRL Issuers;
- OCSP Responders;
- Repository Authorities (e.g., a Directory);
- Time-Stamping Authorities;
- Time-Marking Authorities; and
- Signature Policy Issuers.

Certification Authorities provide users with public key certificates and 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. Signature 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 signer's and verifier's point of view.

Further information on signature policies is provided in:

- TR 102 038 [TR102038];
- Sections 5.8.1, C.1, and C.3.1 of the present document;
- RFC 3125 [RFC3125]; and
- TR 102 272 [TR102272].

#### 4.3. Electronic Signature Formats

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

##### 4.3.1. CADES Basic Electronic Signature (CADES-BES)

A CADES Basic Electronic Signature (CADES-BES), in accordance with the present document, 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; and
- The digital signature value computed on the user data and, when present, on the signed attributes, as defined in CMS (RFC 3852 [4]).

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

- a collection of additional signed attributes; and
- a collection of optional unsigned attributes.

The mandatory signed attributes are:

- Content-type. It is defined in RFC 3852 [4] and specifies the type of the EncapsulatedContentInfo value being signed. Details are provided in Section 5.7.1 of the present document. Rationale for its inclusion is provided in Annex C.3.7;

- 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 Section 5.7.2;
- ESS signing-certificate OR ESS signing-certificate-v2. 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 a digest algorithm. The ESS signing-certificate-v2 attribute is defined in "ESS Update: Adding CertID Algorithm Agility", RFC 5035 [15], and allows for the use of any digest algorithm. A CADES-BES claiming compliance with the present document must include one of them. Section 5.7.3 provides the details of these attributes. Rationale for its inclusion is provided in Annex C.3.3.

Optional signed attributes may be added to the CADES-BES, including optional signed attributes defined in CMS (RFC 3852 [4]), ESS (RFC 2634 [5]), and the present document. Listed below are optional attributes that are defined in Section 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 Section 5.9.1. Annex C.3.6 provides the rationale.
- content-hints: as defined in ESS (RFC 2634 [5]), provides information that describes the innermost signed content of a multi-layer message where one content is encapsulated in another. Section 5.10.1 provides the specification details. Annex C.3.8 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. Section 5.10.1 provides the specification details. Annex C.3.9 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. Section 5.10.2 provides the specification details.
- 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. Section 5.11.1 provides the specification details. Annex C.3.2 provides the rationale.

- `signer-location`: this attribute is defined by the present document. It allows the signer to indicate the place where the signer purportedly produced the signature. Section 5.11.2 provides the specification details. Annex 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. Section 5.11.3 provides the specification details. Annex 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. Section 5.11.4 provides the specification details. Annex C.3.6 provides the rationale.

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

- `CounterSignature`, as defined in CMS (RFC 3852 [4]); it can be incorporated wherever embedded signatures (i.e., a signature on a previous signature) are needed. Section 5.9.2 provides the specification details. Annex C.5 in Annex C provides the rationale.

The structure of the CADES-BES is illustrated in Figure 1.

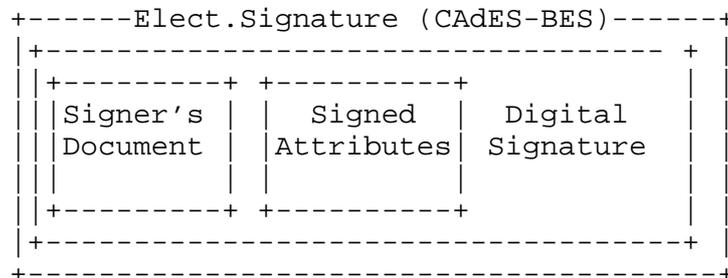


Figure 1: Illustration of a CADES-BES

The signer's conformance requirements of a CADES-BES are defined in Section 8.1.

NOTE: The CADES-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 Section 4.4.2).

The CADES-BES satisfies the legal requirements for electronic signatures, as defined in the European Directive on Electronic Signatures, (see Annex C for further discussion on the relationship of the present document to the Directive). It provides basic authentication and integrity protection.

The semantics of the signed data of a CADES-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. However, further information on signature policies is provided in TR 102 038 [TR102038], RFC 3125 [RFC3125], and Sections 5.8.1, C.1, and C.3.1 of the present document.

#### 4.3.2. CADES Explicit Policy-based Electronic Signatures (CADES-EPES)

A CADES Explicit Policy-based Electronic Signature (CADES-EPES), in accordance with the present document, extends the definition of an electronic signature to conform to the identified signature policy.

A CADES Explicit Policy-based Electronic Signature (CADES-EPES) incorporates a signed attribute (sigPolicyID attribute) indicating the signature policy that shall be used to validate the electronic signature. This signed attribute is protected by the signature. The signature may also have other signed attributes required to conform to the mandated signature policy.

Section 5.7.3 provides the details on the specification of signature-policy-identifier attribute. Annex 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 [TR102038] and Sections 5.8.1, C.1, and C.3.1 of the present document.

The structure of the CADES-EPES is illustrated in Figure 2.

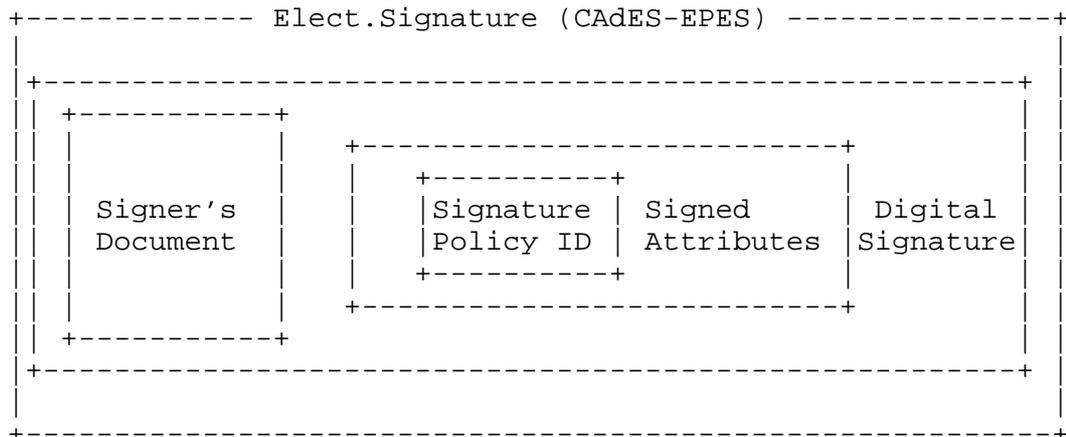


Figure 2: Illustration of a CADES-EPES

The signer's conformance requirements of CADES-EPES are defined in Section 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, otherwise 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-identifier signed attribute 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 online 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 CADES-BES and CADES-EPES, leading to electronic signature formats that include validation data. The sections below summarize these formats and their most relevant characteristics.

#### 4.4.1. Electronic Signature with Time (CADES-T)

An electronic signature with time (CADES-T), in accordance with the present document, is when there exists trusted time associated with the ES.

The trusted time may be provided by:

- a time-stamp attribute as an unsigned attribute added to the ES;  
and
- a time-mark of the ES provided by a Trusted Service Provider.

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

A time-mark provided by a Trusted Service would have a 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 or a time-marked BES/EPES, forming the CADES-T are illustrated in Figure 3.





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 party 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.

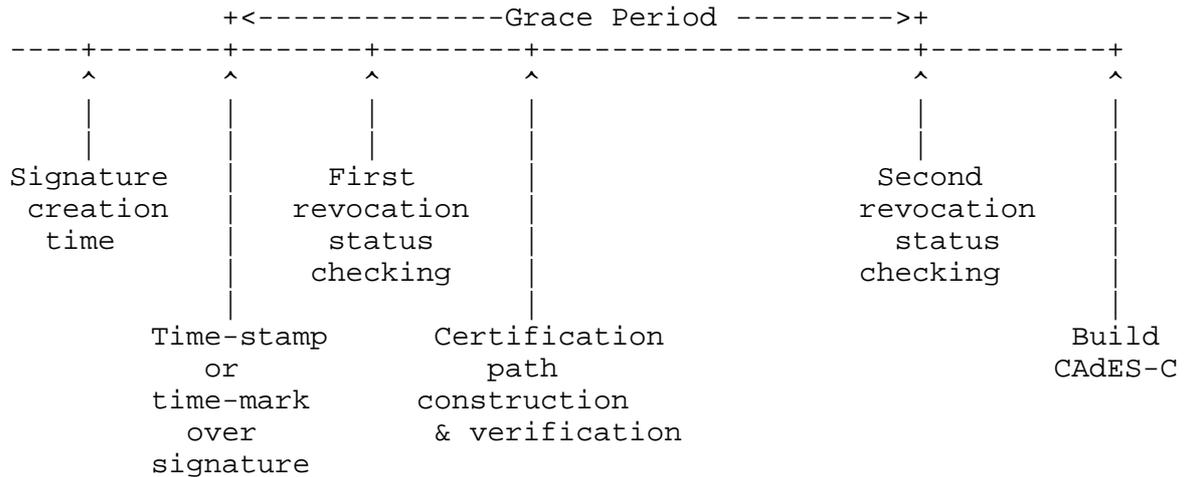


Figure 5: Illustration of a grace period

NOTE 7: CWA 14171 [CWA14171] specifies a signature validation process using CADES-T, CADES-C, and a grace period. Annex B provides example validation processes. Annex C.4 provides additional information about applying grace periods during the validation process.

The verifier’s conformance requirements are defined in Section 8.3 for time-stamped CADES-C, and Section 8.4 for time-marked CADES-C. The present document only defines conformance requirements for the verifier up to an ES with Complete validation data (CADES-C). This means that none of the extended and archive forms of electronic signatures, as defined in Sections 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

CADES-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











- 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 CADES-T is being used, the certificate from the TSU that has issued the time-stamp token in the CADES-T format is not revoked at the time of arbitration;
- when time-marking in the CADES-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 CADES-C was built has not been broken at the time the arbitration is performed; and
- if the signature policy can be explicitly 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; or
- 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 section 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 an 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 that it uses and that 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 be 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: If the content type is id-data, it is recommended that the content be encoded using MIME, and that the MIME type is used to identify the presentation format of the data. See Annex F.1 for an example of using MIME to identify the encoding type.

### 5.3. Signed-data Content Type

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

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

The identification of a signer's certificate used to create the signature is always signed (see Section 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 that 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 :

- When it is present, this allows the signed data to be encapsulated in the SignedData structure, which then contains both the signed data and the signature. However, the signed data may only be accessed by a verifier able to decode the ASN.1 encoded SignedData structure.
- When it is missing, this allows the signed data to be sent or stored separately from the signature, and the SignedData structure only contains the signature. It is, in the case of the signature, only the data that is signed that needs to be stored and distributed 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 Annex 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]), but the `signedAttrs` field shall contain the following attributes:

- `content-type`, as defined in Section 5.7.1; and
- `message-digest`, as defined in Section 5.7.2;
- `signing-certificate`, as defined in Section 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 must be the signer's public key, which shall be verified as correct using the signing certificate reference attribute containing a reference to the signing certificate, i.e., when `SigningCertificateV2` from RFC 5035 [16] or `SigningCertificate` from ESS [5] is used, the public key from the first certificate identified in the sequence of certificate identifiers from `SigningCertificate` must be the key used to verify the digital signature.

### 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 `content-type` attribute indicates the type of the signed content. The syntax of the `content-type` attribute type is as defined in CMS (RFC 3852 [4]) Section 11.1.

NOTE 1: As stated in RFC 3852 [4] , the content-type attribute must have its value (i.e., ContentType) equal to the eContentType of the EncapsulatedContentInfo value being signed.

NOTE 2: For implementations supporting signature generation, if the content-type attribute is id-data, then it is recommended that the eContent be encoded using MIME. For implementations supporting signature verification, if the signed data (i.e., eContent) is MIME-encoded, then the OID of the content-type attribute must be id-data. In both cases, the MIME content-type(s) must be used to identify the presentation format of the data. See Annex F for further details about the use of MIME.

#### 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 ESS-signing-certificate-v2 attribute.

These attributes shall contain a reference to the signer's certificate; they are designed to prevent simple substitution and reissue 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 for 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, defined in ESS [5], must be used if the SHA-1 hashing algorithm is used.
- The ESS signing-certificate-v2 attribute, defined in "ESS Update: Adding CertID Algorithm Agility", RFC 5035 [15], which shall be used when other hashing algorithms are to be used.

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 trust anchor.

### 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 the 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.

If present, the issuerAndSerialNumber in SignerIdentifier field of the SignerInfo shall match the issuerSerial field present in ESSCertID. In addition, the certHash from ESSCertID shall match the SHA-1 hash of the certificate. 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 Section 5.8.3.

### 5.7.3.2. ESS signing-certificate-v2 Attribute Definition

The ESS signing-certificate-v2 attribute is similar to the ESS signing-certificate defined above, except that this attribute can be used with hashing algorithms other than SHA-1.

The syntax of the signing-certificate-v2 attribute type of the ES is as defined in "ESS Update: Adding CertID Algorithm Agility", RFC 5035 [15], and further qualified in the present document.

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

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

The object identifier for this attribute is:

```
id-aa-signingCertificateV2 OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9)
  smime(16) id-aa(2) 47 }
```

If present, the issuerAndSerialNumber in SignerIdentifier field of the SignerInfo shall match the issuerSerial field present in ESSCertIDv2. In addition, the certHash from ESSCertIDv2 shall match the hash of the certificate computed using the hash function specified in the hashAlgorithm 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 1: 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 Section 5.8.3.

NOTE 2: RFC 3126 was using the other signing-certificate attribute (see Section 5.7.3.3) for the same purpose. Its use is now deprecated, since this structure is simpler.

### 5.7.3.3. Other signing-certificate Attribute Definition

RFC 3126 was using the other signing-certificate attribute as an alternative to the ESS signing-certificate when hashing algorithms other than SHA-1 were being used. Its use is now deprecated, since the structure of the signing-certificate-v2 attribute is simpler. Its description is however still present in this version for backwards compatibility.

```
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 CADES-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, and 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 the signature-policy-identifier attribute:

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

signature-policy-identifier attribute values have ASN.1 type SignaturePolicyIdentifier:

```
SignaturePolicyIdentifier ::= CHOICE {
    signaturePolicyId          SignaturePolicyId,
    signaturePolicyImplied     SignaturePolicyImplied
    -- not used in this version
}
```

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

```
SignaturePolicyImplied ::= NULL
```

The sigPolicyId field contains an object-identifier that 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. The hashValue within the sigPolicyHash may be set to zero to indicate that the policy hash value is not known.

NOTE: The use of a zero sigPolicyHash value is to ensure backwards compatibility with earlier versions of the current document. If sigPolicyHash is zero, then the hash value should not be checked against the calculated hash 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
```

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

```
OtherHashValue ::= OCTET STRING
```

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:

- sp-uri: this contains the web URI or URL reference to the signature policy, and
- sp-user-notice: this contains a user notice that 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 appropriate, 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]).

NOTE: RFC 3852 [4] states that dates between January 1, 1950 and December 31, 2049 (inclusive) must be encoded as `UTCTime`. Any dates with year values before 1950 or after 2049 must be encoded as `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 are appropriately 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]).

#### 5.10.2. content-identifier Attribute

The content-identifier attribute provides an identifier for the signed content, for use when a 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 the ES have an 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 on the innermost signed content of a multi-layer message where one content is encapsulated in another.

The syntax of the content-hints attribute type of the ES is 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; and
- 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 of 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 definitions elsewhere; such definitions are outside the scope of the present document.

### 5.11. Additional Optional Attributes Defined in the Present Document

This section defines a number of attributes that may be used to indicate additional information to a verifier:

- a) the type of commitment from the signer, and/or
- b) the claimed location where the signature is performed, and/or

- c) claimed attributes or certified attributes of the signer, and/or
- d) a content time-stamp applied before the content was signed.

#### 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 are defined as part of the signature policy; and
- 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 CommitmentTypeIdIdentifier,
  commitmentTypeQualifier SEQUENCE SIZE (1..MAX) OF
    CommitmentTypeQualifier OPTIONAL }
```

```
CommitmentTypeIdIdentifier ::= 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 meanings:

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; or
- 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 Section 4.1.
```

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

NOTE 2: Attribute and AttributeCertificate are as defined respectively in ITU-T Recommendations X.501 [9] and X.509 [1].

#### 5.11.4. content-time-stamp Attribute

The content-time-stamp attribute is an attribute that 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 the eContent field within encapsContentInfo in the signedData.

For further information and definition of TimeStampToken, see Section 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 Annex 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.

NOTE: Independent signatures may be used to provide independent signatures from different parties with different signed attributes, or to provide multiple signatures from the same party using alternative signature algorithms, in which case the other attributes, excluding time values and signature policy information, will generally be the same.

#### 5.12.2. Embedded Signatures

Multiple embedded signatures (see Annex C.5) are supported using the countersignature unsigned attribute (see Section 5.9.2). Each counter signature is carried in countersignature held as an unsigned attribute to the SignerInfo to which the counter-signature is applied.

NOTE: Counter signatures may be used to provide signatures from different parties with different signed attributes, or to provide multiple signatures from the same party using alternative signature algorithms, in which case the other attributes, excluding time values and signature policy information, will generally be the same.

### 6. Additional Electronic Signature Validation Attributes

This section specifies attributes that contain different types of validation data. These attributes build on the electronic signature specified in Section 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 (CAeS-T); and
- Complete validation data references that comprise the time-stamp of the signature value, 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 (CAeS-C).

NOTE 1: Formats for CAeS-T are illustrated in Section 4.4, and the attributes are defined in Section 6.1.1.

NOTE 2: Formats for CADES-C are illustrated in Section 4.4. The required attributes for the CADES-C signature format are defined in Sections 6.2.1 to 6.2.2; optional attributes are defined in Sections 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 of the extended forms of validation data:

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

NOTE 3: Formats for CADES-X Type 1 and CADES-X Type 2 are illustrated in Sections B.1.2 and B.1.3, respectively.

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

NOTE 4: Formats for CADES-X Long are illustrated in Annex B.1.1.

- CADES-X Long Type 1 or CADES-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 CADES-C as per CADES-X Long.

This section also specifies the data structures used in Archive validation data format (CADES-A) of extended forms:

- Archive form of electronic signature (CADES-A) comprises:
  - the Complete validation data references (CADES-C),
  - the certificate and revocation values (as in a CADES-X Long ),
  - any existing extended electronic signature time-stamps (CADES-X Type 1 or CADES-X Type 2), if present, and
  - 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 Section 6 are unsigned attributes.

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

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

#### 6.1. signature time-stamp Attribute (CADES-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 Section 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 the 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 Section 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 Data References (CADES-C)

An electronic signature with Complete validation data references (CADES-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 CADES-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 Section 6.1.1, or a time-mark operated by a Time-Marking Authority;
- complete-certificate-references, as defined in Section 6.2.1;
- complete-revocation-references, as defined in Section 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 Section 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 Section 5.7.3.3.

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 Section 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 signedData of the relevant time-stamp 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 indicates that the verifier has taken due diligence to gather the available revocation information. The references stored in this attribute can be used to retrieve the referenced information, if not stored in the CMS structure, but somewhere else.

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 {
    crlids      [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 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
}

```

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 produced as contained in the issued OCSP response. Since it may be needed to make the difference between two OCSP responses received within the same second, the hash of the response contained in the `OcspResponsesID` may be needed to solve the ambiguity.

NOTE 1: Copies of the CRL and OCSP responses values may be held using the `revocation-values` attribute defined in Section 6.3.4.

NOTE 2: It is recommended that this attribute be used in preference to the `OtherRevocationInfoFormat` specified in RFC 3852 to maintain backwards compatibility with the earlier version of this specification.

The syntax and semantics of other revocation references are 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 provide time-stamp tokens. In this case, the unsigned attribute shall be added to the `signedData` of the relevant time-stamp token as an `unsignedAttrs` in the `signerInfos` field.

### 6.2.3. attribute-certificate-references Attribute Definition

This attribute is only used when a 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 Section 5.7.3.3.

NOTE: Copies of the certificate values may be held using the certificate-values attribute defined in Section 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 (CAvES-X)

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

### 6.3.1. Time-Stamped Validation Data (CADES-X Type 1 or Type 2)

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

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

### 6.3.2. Long Validation Data (CADES-X Long, CADES-X Long Type 1 or 2)

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

- certificate-values, as defined in Section 6.3.3; and
- revocation-values, as defined in Section 6.3.4.

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

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

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

NOTE 1: The CADES-X-Long signature provides long-term proof of the validity of the signature for as long as the CA keys, CRL Issuers keys, and OCSP responder keys are not compromised and are resistant to cryptographic attacks.

NOTE 2: As long as the time-stamp data remains valid, the CADES-X Long Type 1 and the CADES-X Long Type 2 provide 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 has 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: CADES-X Long, ES X-Long Type 1, and CADES-X Long Type 2; see Annex 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 Section 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 Section 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 signedData of the relevant time-stamp 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: CADES-X Long, ES X-Long Type 1, and CADES-X Long Type 2; see Annex 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.

NOTE: It is recommended that this attribute be used in preference to the OtherRevocationInfoFormat specified in RFC 3852 to maintain backwards compatibility with the earlier version of this specification.

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,
    ocspsVals        [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) are 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 Section 7.2. (which is as defined in ITU-T Recommendation X.509 [1]).

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

This attribute may include the values of revocation data including CRLs and OCSPs 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 signedData of the relevant time-stamp token.

### 6.3.5. CADES-C-time-stamp Attribute Definition

This attribute is used to protect against CA key compromise.

This attribute is used for the time-stamping of the complete electronic signature (CADES-C). It is used in the following forms of extended electronic signature; CADES-X Type 1 and CADES-X Long Type 1; see Annex B.1.2 for an illustration of this form of electronic signature.

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

NOTE 1: It is recommended that the attributes being time-stamped be encoded in DER. If DER is not employed, then the binary encoding of the ASN.1 structures being time-stamped should be preserved to ensure that the recalculation of the data hash is consistent.

NOTE 2: Each attribute is included in the hash with the attrType and attrValues (including type and length) but without the type and length of the outer SEQUENCE.

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

```
ESCTimeStampToken ::= TimeStampToken
```

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

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

- complete-revocation-references attribute.

For further information and definition of the TimeStampToken, see Section 7.4.

#### 6.3.6. time-stamped-certs-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 Annex 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 and/or CRLs to protect against certain CA compromises. Its syntax is as follows:

NOTE 1: It is recommended that the attributes being time-stamped be encoded in DER. If DER is not employed, then the binary encoding of the ASN.1 structures being time-stamped should be preserved to ensure that the recalculation of the data hash is consistent.

NOTE 2: Each attribute is included in the hash with the attrType and attrValues (including type and length) but without the type and length of the outer SEQUENCE.

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

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

The attribute value has the ASN.1 syntax TimestampedCertsCRLs:

```
TimestampedCertsCRLs ::= TimeStampToken
```

The value of the messageImprint field within the 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 (CADES-C):

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

#### 6.4. Archive Validation Data

Where an electronic signature is required to last for a very long time, and 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, 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 the electronic signature (CADES-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 CADES-BES or CADES-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-archiveTimestampV2 OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
smime(16) id-aa(2) 48 }
```

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

```
ArchiveTimeStampToken ::= TimeStampToken
```

The value of the messageImprint field within TimeStampToken shall be a hash of the concatenation of:

- the encapContentInfo element of the SignedData sequence;
- any external content being protected by the signature, if the eContent element of the encapContentInfo is omitted;
- the Certificates and crls elements of the SignedData sequence, when present, and;
- all data elements in the SignerInfo sequence including all signed and unsigned attributes.

NOTE 1: An alternative archiveTimestamp attribute, identified by an object identifier { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 27, is defined in prior versions of TS 101 733 [TS101733] and in RFC 3126.

The archiveTimestamp attribute, defined in versions of TS 101 733 prior to 1.5.1 and in RFC 3126, is not compatible with the attribute defined in the current document. The archiveTimestamp attribute, defined in versions 1.5.1 to 1.6.3 of TS 101 733, is compatible with the current document if the content is internal to encapContentInfo. Unless the version of TS 101 733 employed by the signing party is known by all recipients, use of the archiveTimestamp attribute defined in prior versions of TS 101 733 is deprecated.

NOTE 2: Counter signatures held as countersignature attributes do not require independent archive time-stamps, as they are protected by the archive time-stamp against the containing SignedData structure.

NOTE 3: Unless DER is used throughout, it is recommended that the binary encoding of the ASN.1 structures being time-stamped be preserved when being archived to ensure that the recalculation of the data hash is consistent.

NOTE 4: The hash is calculated over the concatenated data elements as received/stored, including the Type and Length encoding.

NOTE 5: Whilst it is recommended that unsigned attributes be DER encoded, it cannot generally be so guaranteed except by prior arrangement. For further information and definition of TimeStampToken, see Section 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 6: 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 stored elsewhere; or
- Adding the certificate-values attribute and the revocation-values attribute of the TSP as an unsigned attribute within TimeStampToken, when the required information is stored 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].

### 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].

### 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 profiled in ETSI TS 101 861 [TS101861].

### 7.5. Name and Attribute Formats

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

NOTE: The name used by the signer, held as the subject in the signer's certificate, is 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 Certification Authorities, Attribute Authorities, and Time-Stamping Authorities shall use distinguished names in the subject field of their certificate.

The distinguished name 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, which 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 the signer's attribute that may be included in public key certificates.

NOTE: The 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 Section 7.6.

## 7.6. AttributeCertificate

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

## 8. Conformance Requirements

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

For implementations supporting signature verification, 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 the 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 Sections 5.1 and 5.2);
- CMS SignedData, as defined in RFC 3852 [4], with the version set to 3 and at least one SignerInfo present (see Sections 5.3 to 5.6);
  - The following CMS attributes, as defined in RFC 3852 [4]:
    - content-type; this shall always be present (see Section 5.7.1); and
    - message-digest; this shall always be present (see Section 5.7.2).
- One of the following attributes, as defined in the present document:
  - signing-certificate: as defined in Section 5.7.3.1; or
  - signing-certificate v2 : as defined in Section 5.7.3.2.

NOTE: RFC 3126 was using the other signing-certificate attribute (see Section 5.7.3.3). Its use is now deprecated, since the structure of the signing-certificate v2 attribute is simpler than the other signing-certificate attribute.

## 8.2. CADES-Explicit Policy-based Electronic Signature

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

- The following attributes, as defined in Section 5.9:
  - signature-policy-identifier; this shall always be present (see Section 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 Section 8.1;
- signature-time-stamp attribute, as defined in Section 6.1.1;
- complete-certificate-references attribute, as defined in Section 6.2.1;
- complete-revocation-references attribute, as defined in Section 6.2.2;
- Public Key Certificates, as defined in ITU-T Recommendation X.509 [1] (see Section 8.1); and
- either of:
  - Certificate Revocation Lists, as defined in ITU-T Recommendation X.509 [1] (see Section 8.2); or
  - Online Certificate Status Protocol, as defined in RFC 2560 [3] (see Section 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 Section 8.1;

- complete-certificate-references attribute, as defined in Section 6.2.1;
- complete-revocation-references attribute, as defined in Section 6.2.2;
- a record of the electronic signature and the time when the signature was first validated, using the referenced certificates and revocation information, must be maintained, such that records cannot be undetectably modified;
- Public Key Certificates, as defined in ITU-T Recommendation X.509 [1] (see Section 8.1); and
  - either of:
    - Certificate Revocation Lists, as defined in ITU-T Recommendation X.509 [1] (see Section 8.2); or
    - online Certificate Status Protocol, as defined in RFC 2560 [3] (see Section 8.3).

## 9. References

### 9.1. Normative References

- [1] ITU-T Recommendation X.509 (2000)/ISO/IEC 9594-8 (2001): "Information technology - Open Systems Interconnection - The Directory: Public key and Attribute Certificate framework".
- [2] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.
- [3] Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 2560, June 1999.
- [4] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.
- [5] Hoffman, P., Ed., "Enhanced Security Services for S/MIME", RFC 2634, June 1999.
- [6] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

- [7] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, August 2001.
- [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] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, August 2002.
- [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] Farrell, S. and R. Housley, "An Internet Attribute Certificate Profile for Authorization", RFC 3281, April 2002.
- [14] ITU-T Recommendation X.208 (1988): "Specification of Abstract Syntax Notation One (ASN.1)".
- [15] Schaad, J., "Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility", RFC 5035, August 2007.
- [16] ITU-T Recommendation X.690 (2002): "Information technology ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

## 9.2. Informative References

- [EUDirective] Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a community framework for Electronic Signatures.
- [TS101733] ETSI Standard TS 101 733 V.1.7.3 (2005-06) Electronic Signature Formats.
- [TS101861] ETSI TS 101 861: "Time stamping profile".

- [TS101903] ETSI TS 101 903: "XML Advanced Electronic Signatures (XAdES)".
- [TR102038] ETSI TR 102 038: "Electronic Signatures and Infrastructures (ESI); XML format for signature policies".
- [TR102272] ETSI TR 102 272 V1.1.1 (2003-12). "Electronic Signatures and Infrastructures (ESI); ASN.1 format for signature policies".
- [RFC2479] Adams, C., "Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP-GSS-API)", RFC 2479, December 1998.
- [RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.
- [RFC3125] Ross, J., Pinkas, D., and N. Pope, "Electronic Signature Policies", RFC 3125, September 2001.
- [RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.
- [RFC3494] Zeilenga, K., "Lightweight Directory Access Protocol version 2 (LDAPv2) to Historic Status", RFC 3494, March 2003.
- [RFC3851] Ramsdell, B., Ed., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.
- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, September 2005.
- [RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.
- [RFC4523] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP) Schema Definitions for X.509 Certificates", RFC 4523, June 2006.

- [ISO7498-2] ISO 7498-2 (1989): "Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture".
- [ISO9796-2] ISO/IEC 9796-2 (2002): "Information technology - Security techniques - Digital signature schemes giving message recovery - Part 2: Integer factorization based mechanisms".
- [ISO9796-4] ISO/IEC 9796-4 (1998): "Digital signature schemes giving message recovery - Part 4: Discrete logarithm based mechanisms".
- [ISO10118-1] ISO/IEC 10118-1 (2000): "Information technology - Security techniques - Hash-functions - Part 1: General".
- [ISO10118-2] ISO/IEC 10118-2 (2000): "Information technology - Security techniques - Hash-functions - Part 2: Hash-functions using an n-bit block cipher algorithm".
- [ISO10118-3] ISO/IEC 10118-3 (2004): "Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions".
- [ISO10118-4] ISO/IEC 10118-4 (1998): "Information technology - Security techniques - Hash-functions - Part 4: Hash-functions using modular arithmetic".
- [ISO10181-5] ISO/IEC 10181-5: Security Frameworks in Open Systems. Non-Repudiation Framework. April 1997.
- [ISO13888-1] ISO/IEC 13888-1 (2004): "IT security techniques - Non-repudiation - Part 1: General".
- [ISO14888-1] ISO/IEC 14888-1 (1998): "Information technology - Security techniques - Digital signatures with appendix - Part 1: General".
- [ISO14888-2] ISO/IEC 14888-2 (1999): "Information technology - Security techniques - Digital signatures with appendix - Part 2: Identity-based mechanisms".
- [ISO14888-3] ISO/IEC 14888-3 (1998): "Information technology - Security techniques - Digital signatures with appendix - Part 3: Certificate-based mechanisms".

- [ISO15946-2] ISO/IEC 15946-2 (2002): "Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 2: Digital signatures".
- [CWA14171] CWA 14171 CEN Workshop Agreement: "General Guidelines for Electronic Signature Verification".
- [XMLDSIG] XMLDSIG: W3C/IETF Recommendation (February 2002): "XML-Signature Syntax and Processing".
- [X9.30-1] ANSI X9.30-1 (1997): "Public Key Cryptography for the Financial Services Industry - Part 1: The Digital Signature Algorithm (DSA)".
- [X9.30-2] ANSI X9.30-2 (1997): "Public Key Cryptography for the Financial Services Industry - Part 2: The Secure Hash Algorithm (SHA-1)".
- [X9.31-1] ANSI X9.31-1 (1997): "Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry - Part 1: The RSA Signature Algorithm".
- [X9.31-2] ANSI X9.31-2 (1996): "Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry - Part 2: Hash Algorithms".
- [X9.62] ANSI X9.62 (1998): "Public Key Cryptography for the Financial Services Industry - The Elliptic Curve Digital Signature Algorithm (ECDSA)".
- [P1363] IEEE P1363 (2000): "Standard Specifications for Public-Key Cryptography".

ETSI technical specifications can be downloaded free of charge via the Services and Products Download Area at:  
<http://www.etsi.org/WebSite/Standards/StandardsDownload.aspx>

## 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 Annex A.1 using syntax defined in ITU-T Recommendation X.208 [14] has precedence over that defined in Annex 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: ESS Update
```

```
-- RFC 5035 (Adding CertID Algorithm Agility)
```

```
id-aa-signingCertificate, SigningCertificate, IssuerSerial,
id-aa-contentReference, ContentReference, id-aa-contentIdentifier,
ContentIdentifier, id-aa-signingCertificateV2
FROM ExtendedSecurityServices-2006
{ iso(1) member-body(2) us(840) rsadsi(113549)
pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-ess-2006(30) }
```

```
-- 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)}

;

-- Definitions of Object Identifier arcs used in the present document
-- =====

-- OID used referencing electronic signature mechanisms based on
-- the present document for use with the Independent Data Unit
-- Protection (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
-- =====

```

-- OtherSigningCertificate - deprecated

```
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}
```

-- Policy ES Attributes Defined in the present document

-- =====

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

-- 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 }
```

```
SignaturePolicy ::= CHOICE {
    signaturePolicyId      SignaturePolicyId,
    signaturePolicyImplied SignaturePolicyImplied
    -- not used in this version
}
```

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

```
SignaturePolicyImplied ::= NULL
```

```
SigPolicyId ::= OBJECT IDENTIFIER

SigPolicyHash ::= OtherHashAlgAndValue

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

OtherHashValue ::= OCTET STRING

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 attribute

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

CommitmentTypeIndication ::= SEQUENCE {
```

```
commitmentTypeId CommitmentTypeIdIdentifier,  
commitmentTypeQualifier SEQUENCE SIZE (1..MAX) OF  
CommitmentTypeQualifier OPTIONAL}
```

```
CommitmentTypeIdIdentifier ::= OBJECT IDENTIFIER
```

```
CommitmentTypeQualifier ::= SEQUENCE {  
  commitmentTypeIdIdentifier CommitmentTypeIdIdentifier,  
  qualifier ANY DEFINED BY commitmentTypeIdIdentifier }
```

```
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 attribute
```

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

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

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

```
-- Signer-attributes attribute
```

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

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

```
ClaimedAttributes ::= SEQUENCE OF Attribute
```

```
CertifiedAttributes ::= AttributeCertificate
-- as defined in RFC 3281: see Section 4.1
```

```
-- 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}
```

```
ContentTimestamp ::= TimeStampToken
```

```
-- 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}
```

```
SignatureTimeStampToken ::= TimeStampToken
```

```
-- Complete-certificate-references attribute
```

```
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-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}
```

```
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 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}

CertificateValues ::= SEQUENCE OF Certificate

-- Certificate-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}

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
```

```
-- CAeS-C time-stamp attribute
```

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

```
ESCTimeStampToken ::= TimeStampToken
```

```
-- Time-Stamped Certificates and CRLs
```

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

```
TimestampedCertsCRLs ::= TimeStampToken
```

```
-- Archive time-stamp attribute
```

```
id-aa-ets-archiveTimeStampV2 OBJECT IDENTIFIER ::= { iso(1)  
member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)  
smime(16) id-aa(2) 48}
```

```
ArchiveTimeStampToken ::= TimeStampToken
```

```
-- Attribute-certificate-references attribute
```

```
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 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}
```

```
AttributeRevocationRefs ::= SEQUENCE OF CrlOcspRef
```

END

## A.2. Signature Format Definitions Using X.680 ASN.1 Syntax

NOTE: The ASN.1 module defined in Annex A.1 has precedence over that defined in Annex 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: ESS Update

-- RFC 5035 (Adding CertID Algorithm Agility)

```
id-aa-signingCertificate, SigningCertificate, IssuerSerial,
id-aa-contentReference, ContentReference, id-aa-contentIdentifier,
ContentIdentifier, id-aa-signingCertificateV2
  FROM ExtendedSecurityServices-2006
  { iso(1) member-body(2) us(840) rsadsi(113549)
  pkcs(1) pkcs-9(9) smime(16) modules(0) id-mod-ess-2006(30) }
```

-- Internet X.509 Public Key Infrastructure

-- Certificate and CRL Profile: RFC 3280

```
Certificate, AlgorithmIdentifier, CertificateList, Name,
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
```

```
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)}
```

```
-- X.520
```

```
DirectoryString {}
FROM SelectedAttributeTypes
{joint-iso-itu-t ds(5) module(1) selectedAttributeTypes(5) 4}
```

```
;
```

```
-- Definitions of Object Identifier arcs used in the present document
```

```
-- =====
```

```
-- 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

-- OtherSigningCertificate - deprecated

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}

-- 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,
    sigPolicyQualifiers SEQUENCE SIZE (1..MAX) OF
        SigPolicyQualifierInfo OPTIONAL
}

```

```

}

SignaturePolicyImplied ::= NULL

SigPolicyId ::= OBJECT IDENTIFIER

SigPolicyHash ::= OtherHashAlgAndValue

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

OtherHashValue ::= OCTET STRING

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 CommitmentTypeIdentifier,
    commitmentTypeQualifier SEQUENCE SIZE (1..MAX) OF
        CommitmentTypeQualifier OPTIONAL}

```

```

CommitmentTypeIdentifier ::= OBJECT IDENTIFIER

```

```

CommitmentTypeQualifier ::= SEQUENCE {
    commitmentQualifierId COMMITMENT-QUALIFIER.&id,
    qualifier              COMMITMENT-QUALIFIER.&Qualifier OPTIONAL }

```

```

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

```

```

id-cti-ets-proofOfOrigin OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 1}

```

```

id-cti-ets-proofOfReceipt OBJECT IDENTIFIER ::= { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 2}

```

```

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

```

```
id-cti-ets-proofOfSender OBJECT IDENTIFIER ::= { iso(1) member-body(2)
us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) cti(6) 4}
```

```
id-cti-ets-proofOfApproval OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
smime(16) cti(6) 5}
```

```
id-cti-ets-proofOfCreation OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
smime(16) cti(6) 6}
```

-- Signer Location

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

```
SignerLocation ::= SEQUENCE {
-- at least one of the following shall be present
  countryName [0] DirectoryString{maxSize} OPTIONAL,
  -- as used to name a Country in X.520
  localityName [1] DirectoryString{maxSize} OPTIONAL,
  -- as used to name a locality in X.520
  postalAddress [2] PostalAddress OPTIONAL }
```

```
PostalAddress ::= SEQUENCE SIZE(1..6) OF DirectoryString{maxSize}
-- maxSize parametrization as specified in X.683
```

-- 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 Section 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
  }

OcsIdentifier ::= SEQUENCE {
  ocsResponderID      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,
  ocsVals          [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 }

-- CADES-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-archiveTimeStampV2 OBJECT IDENTIFIER ::=
{ iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
smime(16) id-aa(2) 48}

  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

Section 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 provides example validation sequences using the extended formats.

### B.1. Extended Forms of Validation Data

The Complete validation data (CADES-C) described in Section 4.3 and illustrated in Figure 3 may be extended to create electronic signatures with extended validation data. Some electronic signature forms that include extended validation are explained below.

An X-Long electronic signature (CADES-X Long) is the CADES-C with the values of the certificates and revocation information.

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

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

#### B.1.1.1. CADES-X Long

An electronic signature with the additional validation data forming the CADES-X Long form (CADES-X-Long) is illustrated in Figure B.1 and comprises the following:

- CADES-BES or CADES-EPES, as defined in Sections 4.3 , 5.7, or 5.8;
- complete-certificate-references attribute, as defined in Section 6.2.1;
- complete-revocation-references attribute, as defined in Section 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 Section 6.1.1.

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

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

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

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

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





### B.1.3. CADES-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:

- CADES-BES or CADES-EPES, as defined in Sections 4.2, 5.7, or 5.8;
- complete-certificate-references attribute, as defined in Section 6.2.1;
- complete-revocation-references attribute, as defined in Section 6.2.2;
- time-stamped-certs-crls-references attribute, as defined in Section 6.3.6.

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

- signature-time-stamp attribute, as defined in Section 6.1.1.

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

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

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

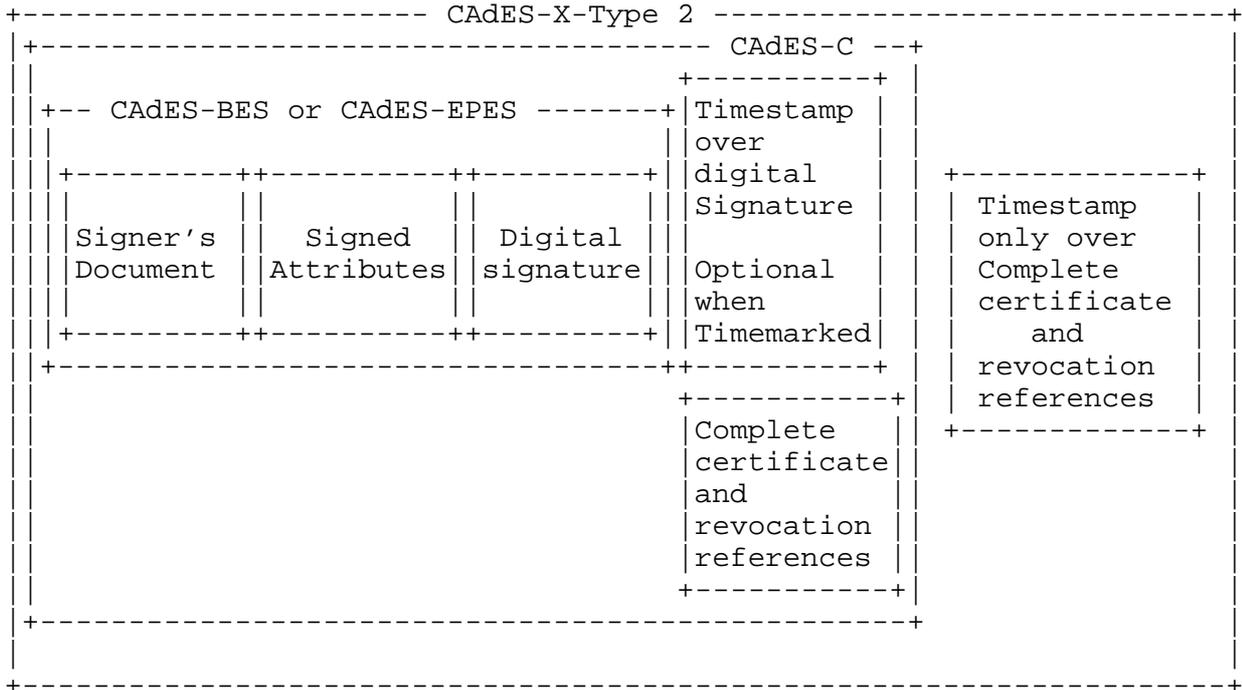


Figure B.3: Illustration of CADES-X Type 2

B.1.4. CADES-X Long Type 1 and CADES-X Long Type 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:

- CADES-BES or CADES-EPES, as defined in Sections 4.3, 5.7, or 5.8;
- complete-certificate-references attribute, as defined in Section 6.2.1;
- complete-revocation-references attribute, as defined in Section 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 Section 6.1.1.

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

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

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

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

Plus one of the following attributes is required:

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

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



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 expire, 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. A 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.



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

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

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

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

At least one of the following two attributes is required:

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

The following attribute is required:

- archive-time-stamp attributes, defined in Section 6.4.1.

Several instances of the 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-time-stamp will itself contain the certificate and revocation information required to validate the archive-time-stamp; this may include the following unsigned attributes:

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



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 certificates 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 the CADES-T and CADES-C processes);
- 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 (5). The end result is called CADES-X Long.

This is illustrated in Figure B.7.

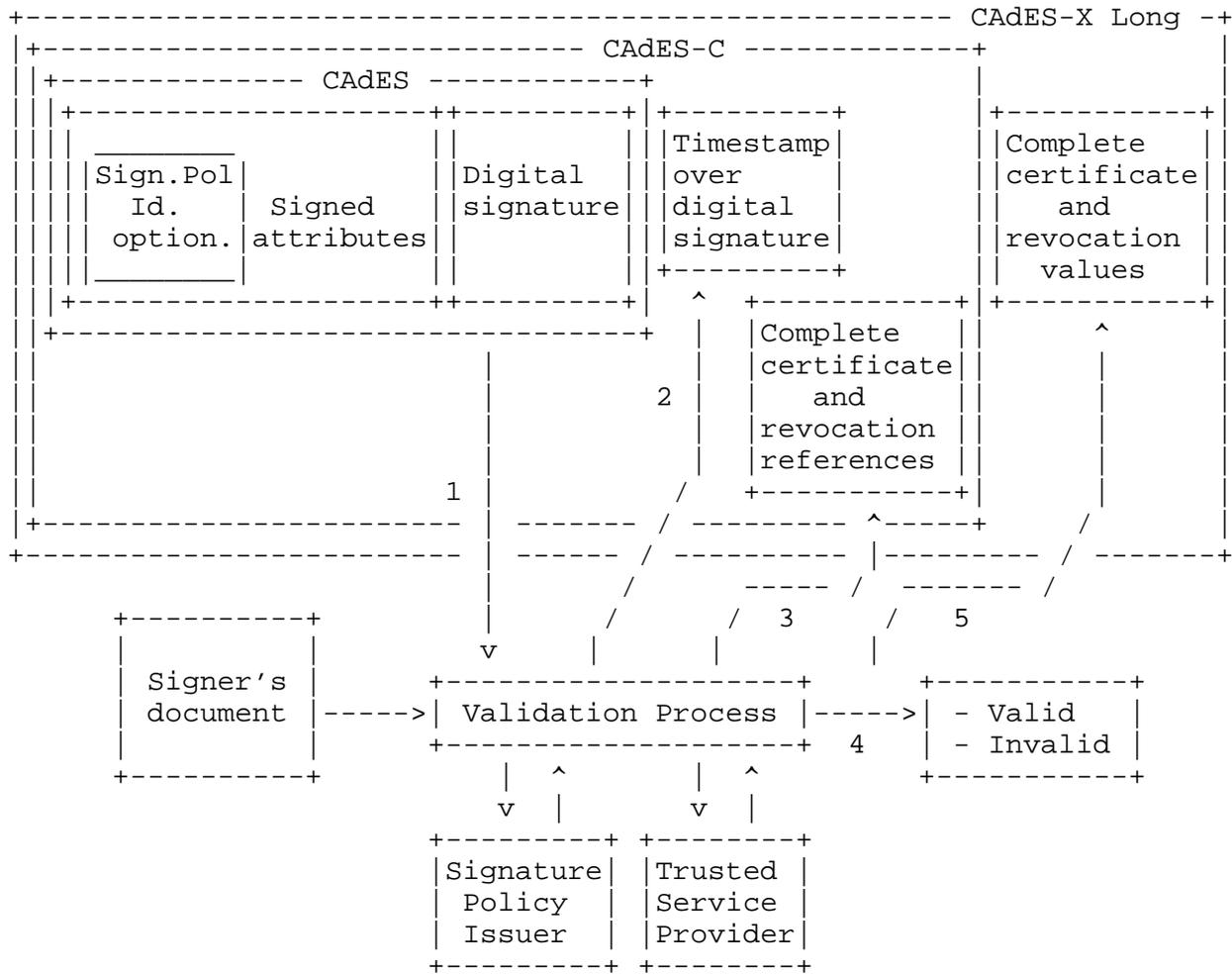


Figure B.7: Illustration of a CADES validation sequence with CADES-X Long

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.

This is illustrated in Figure B.8.



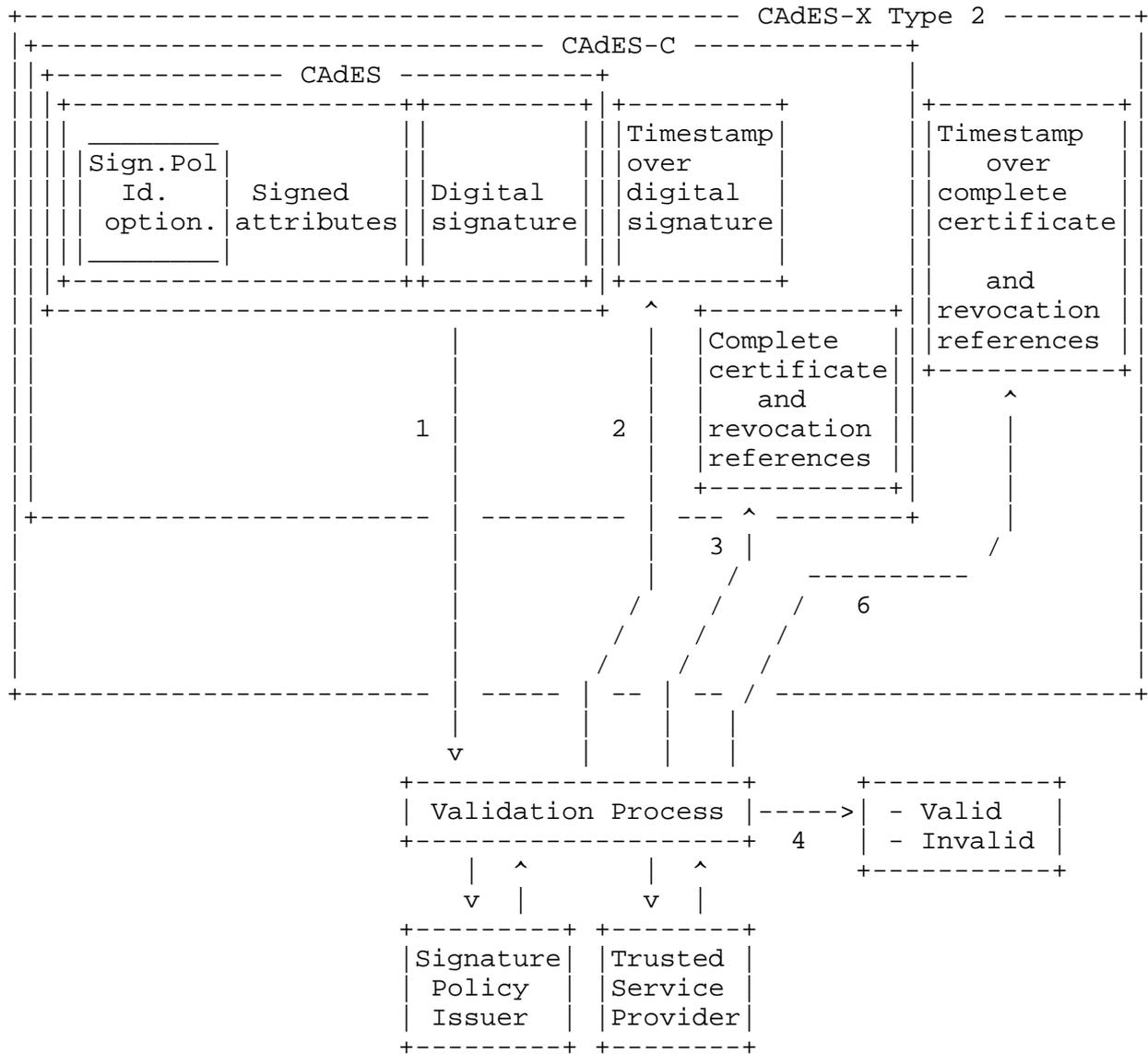


Figure B.9: Illustration of CADES with eXtended validation data CADES-X Type 2

Before the algorithms used in any of the 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).

A CADES-A is illustrated in Figure B.10.



- support counter signatures;
- support multiple signatures.

#### Annex C (Informative): General Description

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

The specification below includes a description of why and when 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 specify the electronic rules for the creation and validation of the electronic signature, also need to be comprehensively defined and in a computer-processable form.

The signature policy thus includes the following:

- rules that apply to technical validation of a particular signature;

- rules that 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 that 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), and;
- 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 that 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" or the eXtensible Mark up Language (XML) may be used. 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 CADES-EPES form of electronic signature; the CADES-EPES mandates a consistent signature policy must be used by both the signer and verifier.

By signing over the Signature Policy Identifier in the CADES-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 CADES-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 type 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 CADES-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 that 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 citizen of a nation 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 that multiple private keys are used, 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, anyone could substitute one certificate for another, and the message would appear to be signed by someone 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 unambiguously identify 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 is 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 the 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 unambiguously identify 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 that is the claimed signing time. When an ES with Time is created (CADES-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.

A further optional attribute is defined in the present document to time-stamp the content and 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 online 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 Section 5.11.4).

#### 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 when data (as opposed to data that has been further signed or encrypted) is encapsulated in the SignedData (indicated by the

eContentType within EncapsulatedContentInfo being set to id-data), further typing information should be used to identify the type of document being signed. This is generally achieved using the MIME content typing and encoding mechanism defined in RFC 2045 [6]). Further information on the use of MIME is given in Annex F.

#### C.3.8. content-hints

The contents-hints attribute provides information on the innermost signed content of a multi-layer message where one content is encapsulated in another. This may be useful if the signed data is itself encrypted.

#### C.3.9. Content Cross-Referencing

When presenting a signed data is in relation to another signed data, it may be important to identify the signed data to which it relates. 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 online certificate status server (for example, obtained through the OCSP protocol).

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

NOTE 2: When validating an electronic signature and checking revocation status information, if a "grace period" is required, it 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 time-stamp 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. Either the signer, the initial verifier, or a subsequent verifier may obtain 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 a CADES-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 so that they can then be easily retrieved and incorporate references to them in the electronic signature itself as a 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, the same comment as for the 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 were 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 this information so that it can be easily retrieved and incorporate references to it in the electronic signature itself as a CADES-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 had been created before the certificate 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 it is the essential component of the ES with Time.

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 be 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 CADES-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 (CADES-X Type 1). This format is suitable to be used with an OCSP response, and it offers the additional advantage of providing 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 (CADES-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 (CADES-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 revocation information references, which include the OCSP response, the time-stamp is placed on the CADES-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 CADES-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 a CADES-X Long Type 1, as defined by the present document.

A CADES-X Long Type 1 is simply the concatenation of a CADES-X Type 1, with a copy of the additional data being referenced.

#### C.4.4.2. Time-Stamping Certificates and Revocation Information References (CADES-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 that 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 be reduced to just one time-stamp per day (i.e., in

the case where 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 CADES-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 CADES-C are covered by one or more time-stamps.

Thus, a CADES-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 cryptanalysis, or improvements in cryptanalytical 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 the Archived Electronic Signature are required.

The potential for a 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 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 CADES-A in the present document.

#### C.4.6. Reference to Additional Data

Using CADES-X Type 1 or CADES-X Type 2 extended validation data, verifiers still need 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 (CADES-C) is adequate. The actual certificates and CRL information reference in the CADES-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 CADES-X Long Type 1 or CADES-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, a predefined 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 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 be 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 described above be used. This may be part of a mutually agreed Signature Validation Policy that is part of an agreed signature policy under which digital signatures 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 a possible period. Thus, the validity period for the TSA's keys should be as long as possible.

Both the CADES-T and the CADES-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 CADES-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 CADES-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 generally the case 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 in which the signatures are applied is important. The capability to sign over signed data shall be provided.

These forms are described in Section 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 occurrences 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 certificates, and cross-certificates can be retrieved from a repository using the Lightweight Directory Access Protocol as defined in RFC 3494 [RFC3494], with the schema defined in RFC 4523 [RFC4523].

#### 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 3494 [RFC3494], with the schema defined in RFC 4523 [RFC4523].

### D.1.3. Online Certificate Status

As an alternative to the use of certificate revocation lists, the status of a 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 4210 [RFC4210].

## Annex E (Informative): Security Considerations

### E.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.

### E.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.

## Annex F (Informative): Example Structured Contents and MIME

## F.1. Use of MIME to Encode Data

The signed content may be structured using MIME (Multipurpose Internet Mail Extensions -- RFC 2045 [6]). Whilst the MIME structure was initially developed for Internet email, it has a number of features that 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:

- providing a means of signalling the type of "object" being carried (e.g., text, image, ZIP file, application data);
- providing a means of associating a file name with an object;
- associating several independent objects (e.g., a document and image) to form a multi-part object;
- handling data encoded in text or binary and, if necessary, re-encoding 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.1.1. 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 pdf document is:

```
Content-Type: application/pdf
Content-Transfer-Encoding: base64
Content-Description: JCFV201.pdf
Content-Disposition: filename="JCFV201.pdf"
```

### F.1.2. Content Encoding

MIME supports a range of mechanisms for encoding 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 that converts characters other than the basic ASCII into an ASCII sequence.

Binary can either be carried:

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

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

### F.1.3. 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 indicating 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
```

```
-----=_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/pdf; name="JCFV201.pdf"
Content-Transfer-Encoding: base64
Content-Description: JCFV201.pdf
Content-Disposition: attachment; filename="JCFV201.pdf"
```

```
OM8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAAAACAAAAAgAAAA
AAAAEAAAtAAAAEAAAD+////AAAAAMAAAAGAAAA////////////////////////////////////
////////////////////////////////////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).

The Content-Type from each part of the S/MIME message indicates the type of content.

## F.2. S/MIME

The specific use of MIME to carry CMS (extended as defined in the present document) secured data is called S/MIME (see [RFC3851]).

S/MIME carries electronic signatures as either:

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

The signed data may be included in the SignedData, which itself may be included in a single S/MIME object. See [RFC3851], Section 3.4.2: "Signing Using application/pkcs7-mime with SignedData" and Figure F.1 hereafter.

or

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

The signed data is not included in the SignedData, and the CMS structure only includes the signature. See [RFC3851], Section 3.4.3: "Signing Using the multipart/signed Format" and Figure F.2 hereafter.

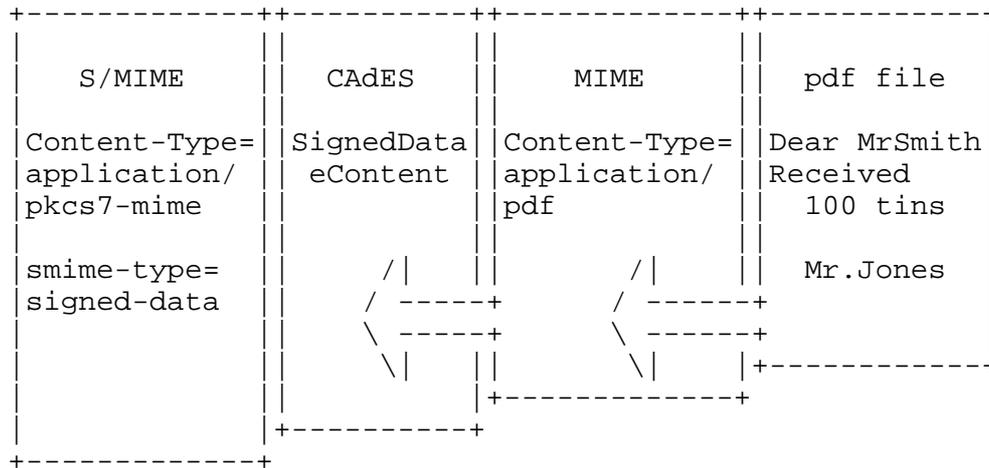


Figure F.1: Signing Using application/pkcs7-mime

F.2.1. Using application/pkcs7-mime

This approach is similar to handling signed data as any other binary file attachment.

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
```

## F.2.2. Using application/pkcs7-signature

CMS also supports an alternative structure where the signature and data being protected are separate MIME objects carried within a single message. In this case, the signed data is not included in the SignedData, and the CMS structure only includes the signature. See [RFC3851], Section 3.4.3: "Signing Using the multipart/signed Format" and Figure F.2 hereafter.

An example of signed data encoded using 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, the signed data passes through the CMS process and is carried as part of a multiple-parts signed MIME structure, as illustrated in Figure F.2. The CMS structure just holds the electronic signature.

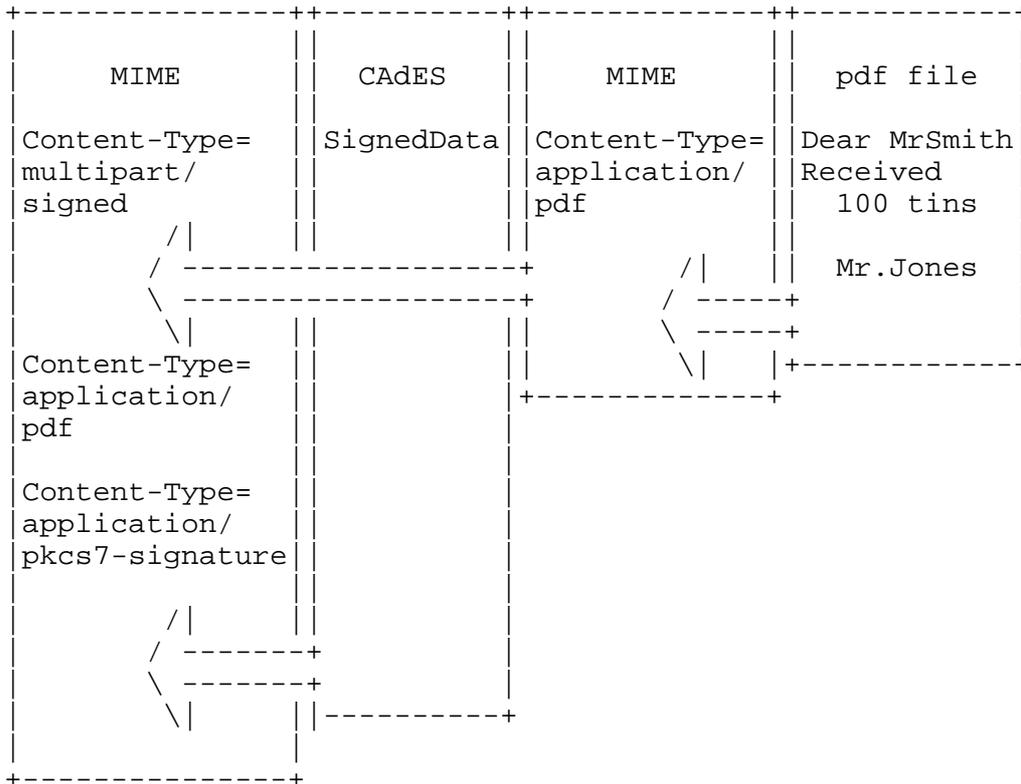


Figure F.2: Signing Using application/pkcs7-signature

This second approach (multipart/signed) has the advantage that the signed data can be decoded by any MIME-compatible system even if it does not recognize CMS-encoded electronic signatures.

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 that has been 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 that meets detailed criteria given in Annex I of the directive and is issued by a "certification-service-provider" that meets requirements given in Annex II of the directive. Such a certificate is referred to as a "qualified certificate";
- it is created by a "device", for which detailed criteria are given in Annex III of 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 standardization elsewhere.

### 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 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; and
- 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 define 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 in close cooperation with actors on the market (business, users, suppliers). The 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. Electronic Signature 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 Section 5 and conformance Section 10.1) is applicable to the proposed class of electronic signature: "qualified electronic signatures fulfilling article 5.1".

With the addition of attributes by the verifier (see Section 6 and conformance Section 10.2) the qualified electronic signature supports long-term validity.

## 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 exist 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. Section 3.1 of RFC 2743 [RFC2743] specifies a mechanism-independent level of encapsulating representation for the initial token of a GSS-API context establishment sequence, incorporating an identifier of the mechanism type to be used on that context 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, with the 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 produced, in December 1998, an RFC (RFC 2479 [RFC2479]) 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 as 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.

Section 2.3.3 of IDUP describes the specific calls needed to handle evidence ("EV" calls). The "EV" group of calls provides a simple, high-level interface to underlying IDUP mechanisms when application developers need to deal with only evidence: 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 the attributes that correspond to a given input token to an application. 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 that 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 that 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 longer considered 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) [ISO10118-1]: "Information technology - Security techniques - Hash-functions - Part 1: General". ISO/IEC 10118-1 contains definitions and describes basic concepts.
- ISO/IEC 10118-2 (1994) [ISO10118-2]: "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) [ISO10118-3]: "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) [ISO10118-4]: "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 outdated.
- RFC 1321 (I 1992): "The MD5 Message-Digest Algorithm". RFC 1321 (informational) specifies the hash-function MD5. Today, MD5 is not recommended for new implementations.
- 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) [X9.30-2]: "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) [X9.31-2]: "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 3447 [RFC3447]. 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:

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

NOTE: RFC 3370 [10] recommends that MD5 not be 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) [P1363]: "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 9796-2 (1997) [ISO9796-2]: "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) [ISO9796-4]: "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 [ISO14888-1]: "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 [ISO14888-2]: "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 [ISO14888-3]: "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) [ISO15946-2]: "Cryptographic techniques based on elliptic curves - Part 2: Digital signatures", 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) [X9.31-1]: "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) [X9.30-1]: "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) [X9.62]: "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 give many examples.

## Annex J (Informative): Guidance on Naming

### J.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".

## J.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 4346 [RFC4346]) with client authentication.

## J.3. Naming Schemes

### J.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. Examples 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.

### J.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 an 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.

#### Acknowledgments

Special thanks to Russ Housley for reviewing the document.

#### Authors' Addresses

Denis Pinkas  
Bull SAS  
Rue Jean-Jaures  
78340 Les Clayes sous Bois CEDEX  
FRANCE  
EMail: Denis.Pinkas@bull.net

Nick Pope  
Thales eSecurity  
Meadow View House  
Long Crendon  
Aylesbury  
Buck  
HP18 9EQ  
United Kingdom  
EMail: nick.pope@thales-esecurity.com

John Ross  
Security & Standards Consultancy Ltd  
The Waterhouse Business Centre  
2 Cromer Way  
Chelmsford  
Essex  
CM1 2QE  
United Kingdom  
EMail: ross@secstan.com

## Full Copyright Statement

Copyright (C) The IETF Trust (2008).

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.

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, THE IETF TRUST 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).

