ETSI TS 101 733 V2.2.1 (2013-04)

Electronic Signatures and Infrastructures (ESI);
CMS Advanced Electronic Signatures (CAdES)

Reference
RTS/ESI-0001733version221

Keywords
CAdES, e-commerce, electronic signature, security

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Electronic Signatures and Infrastructures (ESI).

Introduction

Electronic commerce is emerging as the future way of doing business between companies across local, wide area and global networks. Trust in this way of doing business is essential for the success and continued development of electronic commerce. It is, therefore, important that companies using this electronic means of doing business have suitable security controls and mechanisms in place to protect their transactions and to ensure trust and confidence with their business partners. In this respect the electronic signature is an important security component that can be used to protect information and provide trust in electronic business.

The present 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 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 [i.1]) the validity of the signature.

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

The European Directive on a community framework for Electronic Signatures [i.5] 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 [i.5].

1 Scope

The scope of the present document covers electronic signature formats only, previous versions of the present document covered both Electronic Signature Formats and Electronic Signature Policies. The aspects of Electronic Signature Policies covered by previous versions of the present document are now defined in TR 102 272 [i.2].

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. The present 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/Recommendation ITU-T X.509 [1]: "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks";
- 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 clause 2 for a full set of references.

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

Another document, TS 101 903 [i.3], describes formats for XML Advanced Electronic Signatures (XAdES) built on XMLDSIG.

In addition, the present document identifies other documents that define formats for Public Key Certificates, Attribute Certificates, 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 rational 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 [i.5] 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);
- naming schemes (annex J);
- timestamp hash computation (annex K); and
- changes from previous versions (annex L).

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the reference document (including any amendments) applies.

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

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] Recommendation ITU-T X.509 (2008)/ISO/IEC 9594-8 (2008): "Information technology - Open Systems Interconnection - The Directory: Public-key and Attribute Certificate frameworks".

[2] IETF RFC 3280 (2002): "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[3] IETF RFC 2560 (1999): "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP".

[4] IETF RFC 3852 (2004): "Cryptographic Message Syntax (CMS)".

[5] IETF RFC 2634 (1999): "Enhanced Security Services for S/MIME".

[6] IETF RFC 2045 (1996): "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".

[7] IETF RFC 3161 (2001): "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)".

[8] Recommendation ITU-T X.680 (2008): "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation".

[9] Recommendation ITU-T X.501 (2008)/ISO/IEC 9594-1 (2008): "Information technology - Open Systems Interconnection - The Directory: Models".

[10] IETF RFC 3370 (2002): "Cryptographic Message Syntax (CMS) Algorithms".

[11] Recommendation ITU-T F.1: "Operational provisions for the international public telegram service".

[12] Recommendation ITU-T X.500: "Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services".

[13] IETF RFC 3281 (2002): "An Internet Attribute Certificate Profile for Authorization".

[14] Recommendation ITU-T X.208 (1988): "Specification of Abstract Syntax Notation One (ASN.1)".

NOTE: Recommendation ITU-T X.208 has been withdrawn on 30 October 2002 as it has been superseded by Recommendations ITU-T X.680-683. All known defects in X.208 have been corrected in Recommendations ITU-T X.680-683 (1993) further revised in 1997 and 2002. However, the reference is kept in the current to ensure compatibility with RFC 3852 [4].

[15] IETF RFC 5035 (2007): "Enhanced Security Services (ESS) Update: Adding CertID Algorithm Agility".

[16] IETF RFC 4998 (2007): "Evidence Record Syntax (ERS)".

[17] Void.

2.2 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ISO/IEC 10181-5 (September 1996): "Information technology - Open Systems Interconnection - Security frameworks for open systems: Confidentiality framework".

[i.2] ETSI TR 102 272: "Electronic Signatures and Infrastructures (ESI); ASN.1 format for signature policies".

[i.3] ETSI TS 101 903: "Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES)".

[i.4] ISO 7498-2 (1989): "Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture".

[i.5] Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

[i.6] ISO/IEC 13888-1 (2004): "IT security techniques - Non-repudiation - Part 1: General".

[i.7] ETSI TR 102 038: "TC Security - Electronic Signatures and Infrastructures (ESI); XML format for signature policies".

[i.8] IETF RFC 3125 (2000): "Electronic Signature Policies".

[i.9] ETSI TS 101 861: "Electronic Signatures and Infrastructures (ESI); Time stamping profile".

[i.10] IETF RFC 3494 (2003): "Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2".

[i.11] IETF RFC 4523 (2006): "Lightweight Directory Access Protocol (LDAP). Schema Definitions for X.509 Certificates".

[i.12] IETF RFC 4210 (2005): "Internet X.509 Public Key Infrastructure Certificate Management Protocols".

[i.13] IETF RFC 3851: "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification".

[i.14] IETF RFC 2743 (2000): "Generic Security Service Application Program Interface Version 2, Update 1".

[i.15] IETF RFC 2479 (1998): "Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP-GSS-API)".

[i.16] ISO/IEC 10118-1 (2000): "Information technology - Security techniques - Hash-functions - Part 1: General".

[i.17] ISO/IEC 10118-2 (2000): "Information technology - Security techniques - Hash-functions - Part 2: Hash-functions using an n-bit block cipher".

[i.18] ISO/IEC 10118-3 (2004): "Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions".

[i.19] ISO/IEC 10118-4 (1998): "Information technology - Security techniques - Hash-functions - Part 4: Hash-functions using modular arithmetic".

[i.20] ANSI X9.30-2 (1997): "Public Key Cryptography Using Irreversible Algorithms - Part 2: The Secure Hash Algorithm (SHA-1)".

[i.21] ANSI X9.31-2 (1996): "Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry - Part 2: Hash Algorithms".

[i.22] IETF RFC 3447 (2003): "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1".

[i.23] IEEE 1363 (2000): "Standard Specifications For Public Key Cryptography".

[i.24] ISO/IEC 9796 (all parts): "Information technology - Security techniques - Digital signature schemes giving message recovery".

[i.25] ISO/IEC 9796-2:2002/Amd 1:2008: "Information technology - Security techniques - Digital signature schemes giving message recovery - Part 2: Integer factorization based mechanisms".

[i.26] ISO/IEC 9796-3: "Information technology - Security techniques - Digital signature schemes giving message recovery - Part 3: Discrete logarithm based mechanisms".

[i.27] ISO/IEC 14888-1 (2008): "Information technology - Security techniques - Digital signatures with appendix - Part 1: General".

[i.28] ISO/IEC 14888-2 (2008): "Information technology - Security techniques - Digital signatures with appendix - Part 2: Integer factorization based mechanisms".

[i.29] ISO/IEC 14888-3 (2006): "Information technology - Security techniques - Digital signatures with appendix - Part 3: Discrete logarithm based mechanisms".

[i.30] ISO/IEC 11770-3 (2008): "Information technology - Security techniques - Key management - Part 3: Mechanisms using asymmetric techniques".

[i.31] ANSI X9.30.1 (1997): "Public Key Cryptography Using Irreversible Algorithms - Part 1: The Digital Signature Algorithm (DSA)".

[i.32] ANSI X9.62 (2005): "Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)".

[i.33] IETF RFC 2616 (1999): "Hypertext Transfer Protocol - HTTP/1.1".

[i.34] IETF RFC 4346 (2006): "The TLS Protocol Version 1.1".

[i.35] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[i.36] IETF RFC 3369 (2002): "Cryptographic Message Syntax (CMS)".

[i.37] ISO/IEC 15946-2 (2002): "Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 2: Digital signatures".

[i.38] CWA 14171:2004: "General guidelines for electronic signature verification".

[i.39] FIPS Pub 180-2: "Secure Hash Standard (SHS)".

[i.40] ANSI X9.TR 31 (2005): "Interoperable Secure Key Exchange Key Block Specification for Symmetric Algorithms, Includes Supplement (2009)".

[i.41] ANSI X9.31-1 (1993): "American National Standard, Public-Key Cryptography Using Reversible Algorithms for the Financial Services Industry".

[i.42] FIPS PUB 180-1: "Secure Hash Standard".

[i.43] FIPS PUB 186-2 (2000): "Digital Signature Standard (DSS)".

3 Definitions and abbreviations

3.1 Definitions

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

arbitrator: arbitrate a dispute between a signer and verifier when there is a disagreement on the validity of a digital signature

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

Attribute Authority Revocation List (AARL): revocation list containing a list of references to certificates issued to AAs that are no longer considered valid by the issuing authority

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

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

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

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

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

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

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

NOTE: See ISO 7498-2 [i.4].

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 [i.5] of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.

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

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

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

initial verification: 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 key of a user, together with some other information, rendered unforgeable by digital signature with the private key of the certification authority which issued it

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

Rivest-Shamir-Adleman (RSA): 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: set of rules for the creation and validation of an electronic signature that defines the technical and procedural requirements for electronic signature creation and validation, in order to meet a particular business need, and under which the signature can be determined to be valid

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

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

signer: entity that creates an electronic signature

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

time-marking authority: trusted third party that creates records in an audit trail in order to indicate that a datum existed before a particular point in time

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

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

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

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

valid electronic signature: electronic signature that passes validation

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

verifier: entity that verifies evidence

NOTE 1: See ISO/IEC 13888-1 [i.6].

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
ARL	Authority Revocation List
ASCII	American Standard Code for Information Interchange
ASN	Abstract Syntax Notation

ASN.1 Abstract Syntax Notation 1
ATSv2 archive-time-stamp attribute

NOTE: As defined in clause 6.4.1.

ATSv3 archive-time-stamp-v3 attribute

NOTE: As defined in clause 6.4.2.

BER Basic Encoding Rules
BES Basic Electronic Signature
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-LT CAdES Long Term
CAdES-T CAdES with Time
CAdES-X Long CAdES with Extended Long validation data
CAdES-X CAdES with eXtended validation data
CAdES-XL CAdES –X Long
CARL Certification Authority Revocation List
CEN European Committee for Standardization
CMS Cryptographic Message Syntax
CMS-C Cryptographic Message Syntax with Complete validation data
CMS-T Cryptographic Message Syntax with Time
CMS-X Cryptographic Message Syntax with with eXtended validation data
CORBA Common Object Request Broker Architecture
CRL Certificate Revocation List
CWA CEN Workshop Agreement
DER Distinguished Encoding Rules (for ASN.1)
DH Diffie-Hellman (key agreement protocol)
DL Discrete Logarithm
DSA Digital Signature Algorithm

NOTE: See annex E on cryptographic algorithms.

DSS Digital Signature Standard
EC Elliptic Curve
EC-AMV Elliptic Curve-Agnew Muller Vanstone
EC-DH Elliptic curve Diffie–Hellman (key agreement protocol)
ECDSA Elliptic Curve-Digital Signature Algorithm
EC-NR Elliptic Curve-Nyberg Rueppel
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)
FIPS Federal Information Processing Standard
GSM Global System for Mobile Communications
HTML HyperText Markup Language
HTTP Hypertext Transfer Protocol
IDL Interface Definition Language
IDUP Independent Data Unit Protection
IF Integer Factoring
LT Long-Term
MIME Multipurpose Internet Mail Extensions
MQV Menezes-Qu-Vanstone (key agreement protocol)
NIST National Institute of Standards & Technology
OCSP Online Certificate Status Provider
OID Object IDentifier

OMG	Object Management Group
PKC	Public Key Certificate
PKIX	Public Key Infrastructure using X.509 (IETF Working Group)
RA	Registration Authority
RSA	Rivest-Shamir-Adleman
SHA	Secure Hash Algorithm
SHA-1	Secure Hash Algorithm 1

NOTE: See annex E on cryptographic algorithms.

SHS	Secure Hash Standard
SIM	Subscriber Identity Module
TSA	Time-Stamping Authority
TSP	Trusted Service Provider
TST	Time-Stamp Token
TSU	Time-Stamping Unit
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XAdES	XML Advanced Electronic Signatures
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 clause:

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

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

4.1 Major Parties

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

- the signer;
- the verifier;
- Trusted Service Providers (TSP); 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.
- 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 [i.7];
- clauses 5.8.1, C.1 and C.3.1 of the present document;

- RFC 3125 [i.8]; and
- TR 102 272 [i.2].

4.3 Electronic Signature Formats

The current clause provides an overview for two forms of CMS advanced electronic signature specified in the present document, namely, the 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 clause 5.7.1 of the present document. Rationale for its inclusion is provided in clause 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 clause 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 digest algorithm. The ESS signing-certificate-v2 attribute is defined in "ESS Update: Adding CertID Algorithm Agility", (published as RFC 5035 [15] and allows for the use of any digest algorithm. A CAdES-BES claiming compliance with the present document shall include one of them. Clause 5.7.3 provides the details of these attributes. Rationale for its inclusion is provided in clause 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 clause 5 and have a rationale provided in annex C:

- `Signing-time:` as defined in CMS (RFC 3852 [4]), indicates the time of the signature, as claimed by the signer. Details and short rationale are provided in clause 5.9.1. Clause C.3.6 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. Clause 5.10.1 provides the specification details. Clause 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. Clause 5.10.1 provides the specification details. Clause 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. Clause 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. Clause 5.11.1 provides the specification details. Clause 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. Clause 5.11.2 provides the specification details. Clause C.3.5 provides the rationale.

- `Signer-attributes`: this attribute is defined by the present document. It allows a claimed or certified role to be incorporated into the signed information. Clause 5.11.3 provides the specification details. Clause C.3.4 provides the rationale.

- `content-time-stamp`: this attribute is defined by the present document. It allows a time-stamp token of the data to be signed to be incorporated into the signed information. It provides proof of the existence of the data before the signature was created. Clause 5.11.4 provides the specification details. Clause C.3.6 provides the rationale.

- `mime-type`: this attribute is defined by the present document. It allows a mime-type to be incorporated in the signed information. Clause 5.11.5 provides the specification details. Clause C.3.9 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. Clause 5.9.2 provides the specification details. Clause C.5 provides the rationale.

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

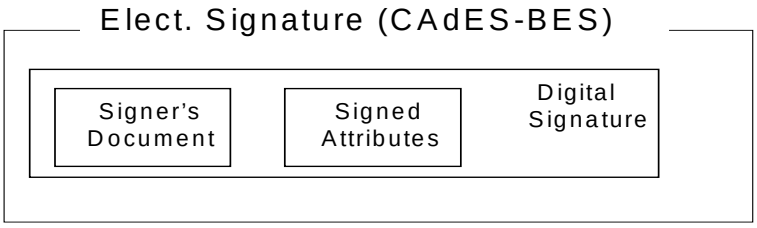


Figure 1: Illustration of a CAdES-BES

The signer's conformance requirements of a CAdES-BES are defined in clause 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 clause 4.4.2).

The CAdES-BES satisfies the legal requirements for electronic signatures, as defined in the European Directive on Electronic Signatures [i.5], (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 [i.7], RFC 3125 [i.8], and clauses 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.

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

Further information on signature policies is provided in TR 102 038 [i.7]; and clauses 5.8.1, C.1 and C.3.1 of the present document.

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

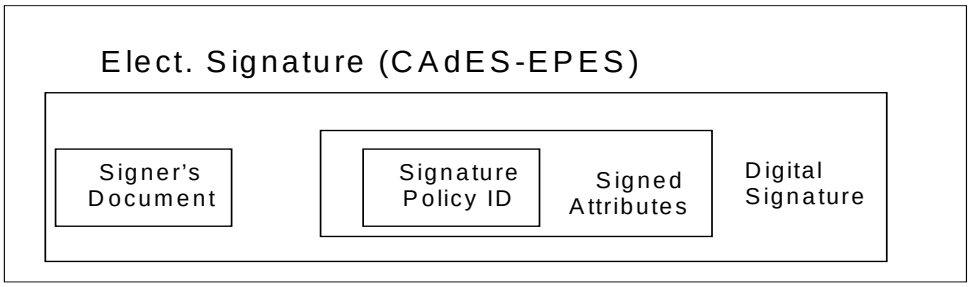


Figure 2: Illustration of a CAdES-EPES

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

4.4 Electronic Signature Formats with Validation Data

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

- Public Key Certificates (PKCs);
- revocation status information for each PKC;
- trusted time-stamps applied to the digital signature, 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. Clauses 4.4.1 to 4.4.5 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 exits trusted time associated with the ES.

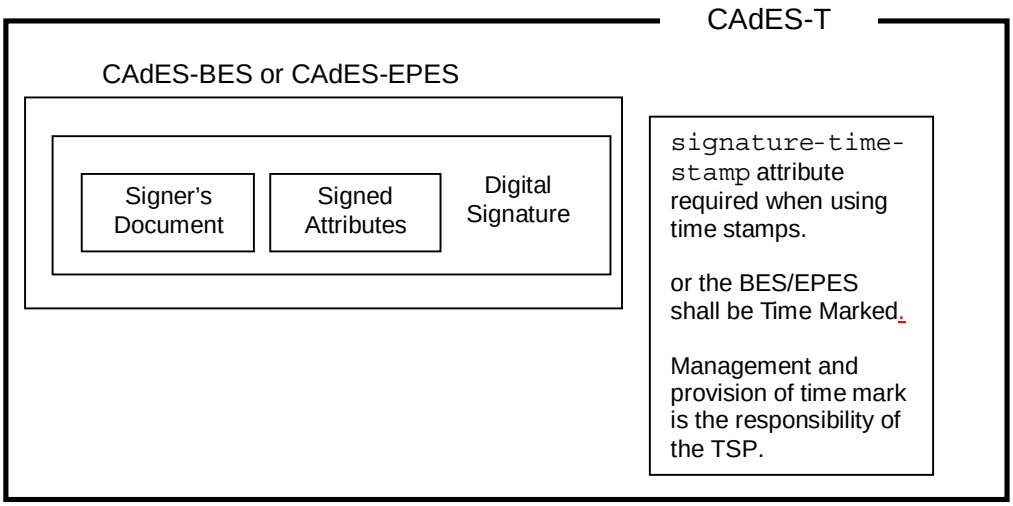
The trusted time may be provided by:

- 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. Clause 6.1.1 provides the specification details. Clause 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 1: A time-stamp token is added to the CAdES-BES or CAdES-EPES as an unsigned attribute.
NOTE 2: Time-stamp tokens that may themselves include unsigned attributes required to validate the time-stamp token, such as the complete-certificate-references and complete-revocation-references attributes, as defined by the present document.

Figure 3: Illustration of CAdES-T formats

4.4.2 ES with Complete Validation Data References (CAdES-C)

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

Clauses C.4.1 to C.4.2 provide rationale on the usage of validation data and when it is suitable to generate the CAdES-C form.

Electronic signatures, with the additional validation data forming the CAdES-C, are illustrated in figure 4.

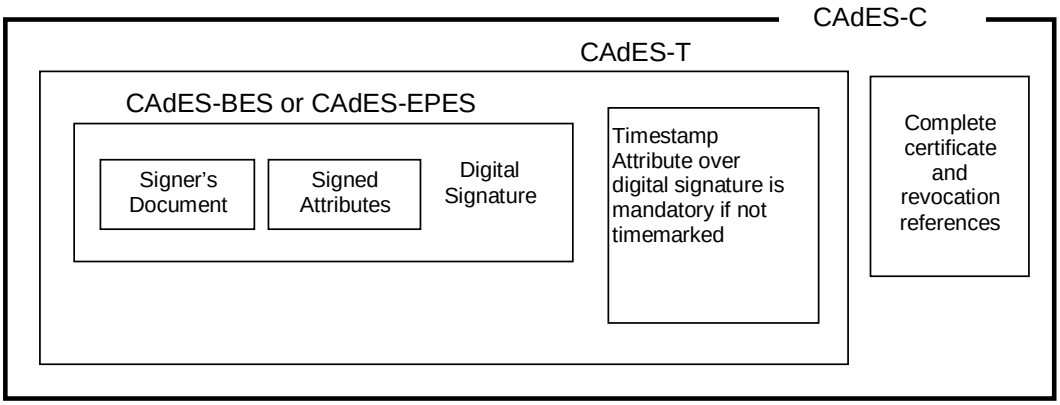


Figure 4: Illustration of CAdES-C format

NOTE 1: The complete certificate and revocation references are added to the CAdES-T as an unsigned attribute.

NOTE 2: As a minimum, the signer will provide the CAdES-BES or, when indicating that the signature conforms to an explicit signing policy, the CAdES-EPES.

NOTE 3: To reduce the risk of repudiating signature creation, the trusted time indication needs to be as close as possible to the time the signature was created. The signer or a TSP could provide the CAdES-T; if not, the verifier should create the CAdES-T on first receipt of an electronic signature because the CAdES-T provides independent evidence of the existence of the signature prior to the trusted time indication.

NOTE 4: A CAdES-T trusted time indication is meant to be created before a certificate has been revoked or expired.

NOTE 5: The signer and TSP could provide the CAdES-C to minimize this risk, and when the signer does not provide the CAdES-C, the verifier should create the CAdES-C when the required component of revocation and validation data become available; this may require a grace period.

NOTE 6: A grace period permits certificate revocation information to propagate through the revocation processes. This period could extend from the time an authorized entity requests certificate revocation to when the information is available for the relying 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.

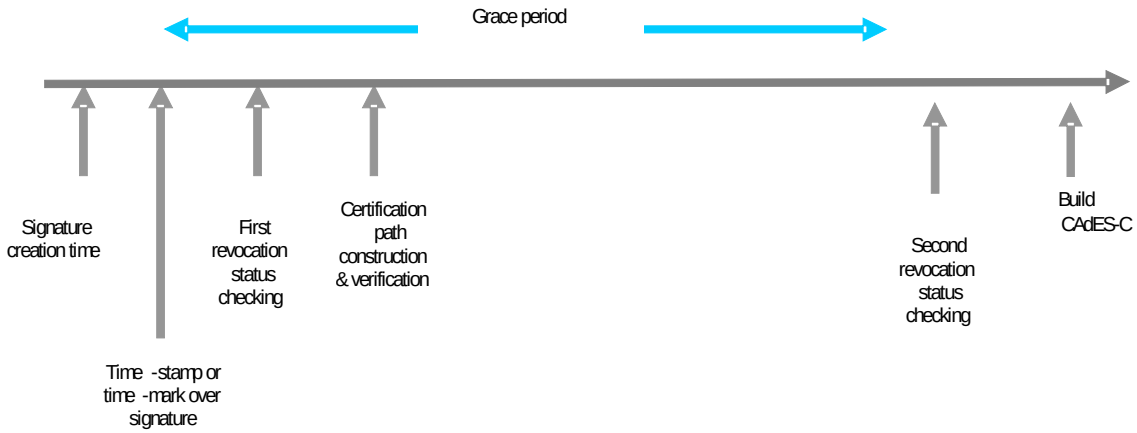


Figure 5: Illustration of a grace period

NOTE 7: CWA 14171 [i.38] specifies a signature validation process using CAdES-T, CAdES-C, and a grace period. Annex B provides example validation processes. Clause C.4 provides additional information about applying grace periods during the validation process.

The verifier's conformance requirements are defined in clause 8.3 for time stamped CAdES-C, and clause 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 signature as defined in clauses 4.4.3 to 4.4.4, need to be implemented to achieve conformance to the present document.

4.4.3 Extended Electronic Signature Formats

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 for preventing some disaster situations that are discussed in annex C. Annex B provides the details of the various extended formats, all the required unsigned attributes for each type, and how they can be used within the electronic signature validation process. Clauses 4.4.3.1 to 4.4.3.4 give an overview of the various forms of extended signature formats in the present document.

4.4.3.1 EXtended Long Electronic Signature (CAdES-X Long)

Extended Long format (**CAdES-X Long**), in accordance with the present document, adds the `certificate-values` and `revocation-values` attributes to the CAdES-C format. The first one contains the whole certificate path required for verifying the signature; the second one contains the CRLs and/OCSP responses required for the validation of the signature. This provides a known repository of certificate and revocation information required to validate a CAdES-C and prevents such information from getting lost. Clauses 6.3.3 and 6.3.4 give specification details. Clause B.1.1 gives details on the production of the format. Clauses C.4.1 to C.4.2 provide the rationale.

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

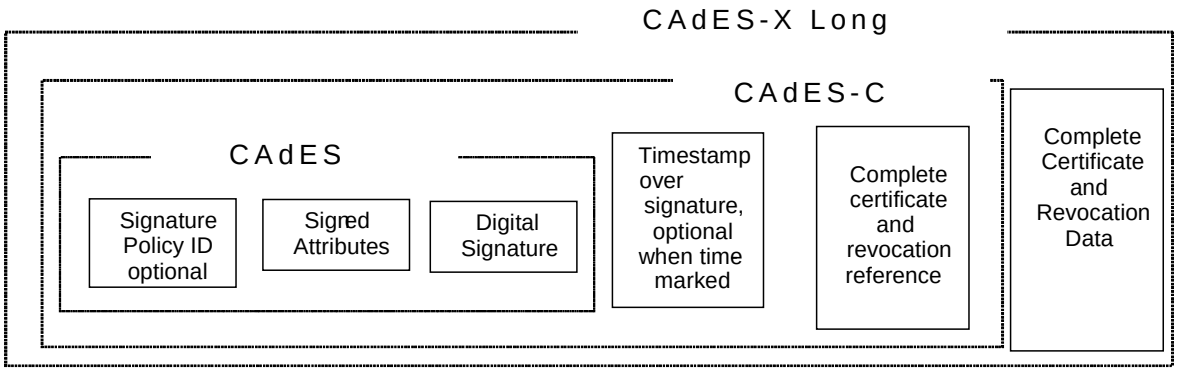


Figure 6: Illustration of CAdES-X-Long

4.4.3.2 EXtended Electronic Signature with Time Type 1 (CAdES-X Type 1)

Extended format with time type 1 (**CAdES-X Type 1**), in accordance with the present document, adds to the CAdES-C-time-stamp attribute, whose content is a time-stamp token on the CAdES-C itself, to the CAdES-C format. This provides an integrity and trusted time protection over all the elements and references. It may protect the certificates, CRLs, and OCSP responses in case of a later compromise of a CA key, CRL key, or OCSP issuer key. Clause 6.3.5 provides the specification details. Clause B.1.2 gives details on the production of the time-stamping process. Clause C.4.4.1 provides the rationale.

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

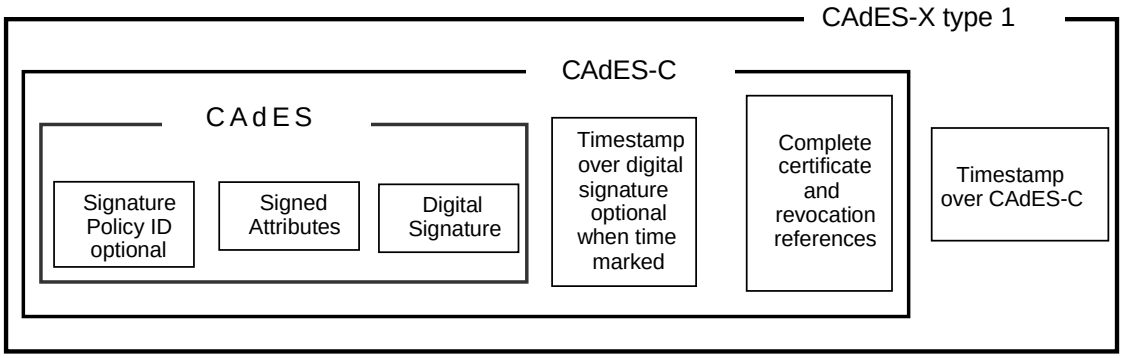


Figure 7: Illustration of CAdES-X Type 1

4.4.3.3 EXtended Electronic Signature with Time Type 2 (CAdES-X Type 2)

Extended format with time type 2 (**CAdES-X Type 2**), in accordance with the present document, adds to the CAdES-C format the `CAdES-C-time-stamped-certs-crls-references` attribute, whose content is a time-stamp token on the certification path and revocation information references. This provides an integrity and trusted time protection over all the references. It may protect the certificates, CRLs and OCSP responses in case of a later compromise of a CA key, CRL key or OCSP issuer key.

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

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

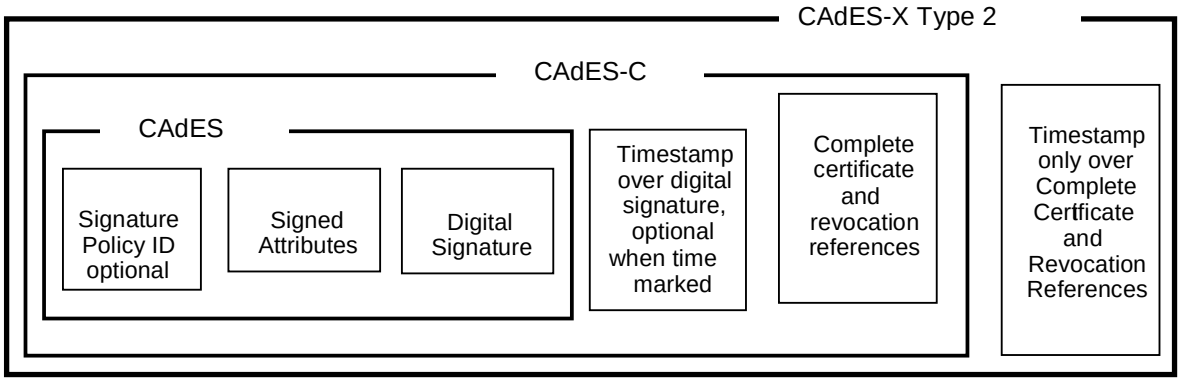


Figure 8: Illustration of CAdES-X Type 2

4.4.3.4 EXtended Long Electronic Signature with Time (CAdES-X Long Type 1 or 2)

Extended Long with Time (**CAdES-X Long Type 1 or 2**), in accordance with the present document, is a combination of **CAdES-X Long** and one of the two former types (**CAdES-X Type 1** and **CAdES-X Type 2**). Clause B.1.4 gives details on the production of the time-stamping process. Clause C.4.8 provides the rationale.

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

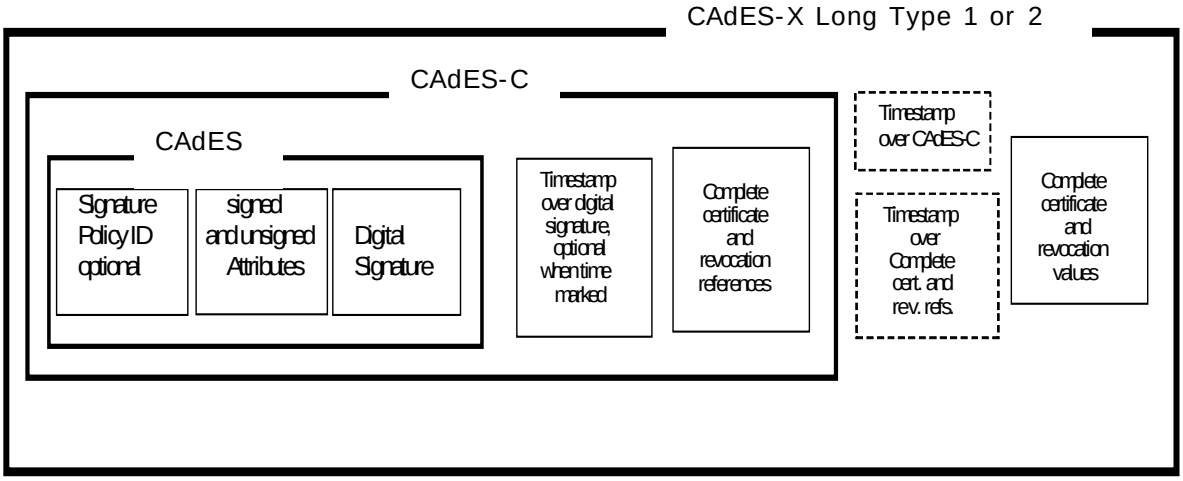


Figure 9: Illustration of CAdES-X Long Type 1 and CAdES-X Long Type 2

4.4.4 Archival Electronic Signature (CAdES-A)

The present document defines two archival forms (CAdES-A) for CAdES signatures, namely: a CAdES-A form with archive-time-stamp (ATSv2) attribute, and a CAdES-A form with the archive-time-stamp-v3 (ATSv3) attribute.

Systems claiming conformance to CAdES-A, as defined in the present document, shall generate archive time-stamps using ATSv3. ATSv2 is included for backward compatibility. Systems may verify the signatures created with ATSv2 and extend their lifetime using ATSv3.

ATSv3 builds on the concepts defined in ATSv2 defined in 6.4.1 and the long-term-validation attribute defined in clause 6.5, ensuring unambiguous processing of the structures within the signature and providing greater flexibility for the incorporation of unsigned attributes (countersignatures, for instance) or parallel signatures, after the signature has been archive time-stamped.

4.4.4.1 CAdES-A with archive-time-stamp (ATSv2) attribute

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

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

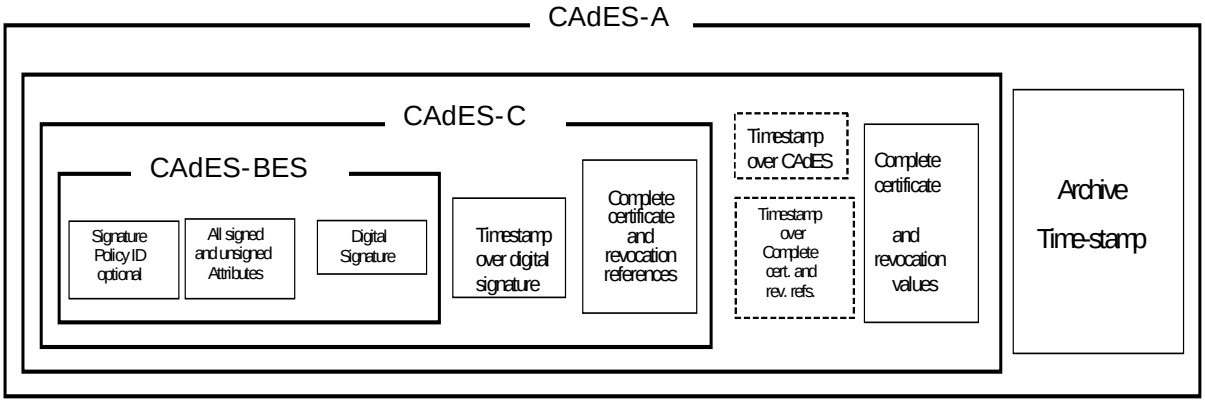


Figure 10: Illustration of CAdES-A

4.4.4.2 CAdES-A with archive-time-stamp (ATSv3) attribute

Archival Form (CAdES-A) with archive-time-stamp-v3 attribute, in accordance with the present document may be built on CAdES-BES, CAdES-EPES, CAdES-T, CAdES-C, CAdES-X, CAdES-XL or CAdES-A by adding one or more archive-time-stamp-v3 attributes. This form is used for archival of long-term signatures. Successive time-stamps protect the whole material against vulnerable hashing algorithms or the breaking of the cryptographic material or algorithms. Clause 6.4 contains the specification details. Clauses C.4.5 and C.4.8 provide the rationale.

The structure of the CAdES-A with ATSv3 form is shown in figure 10A.

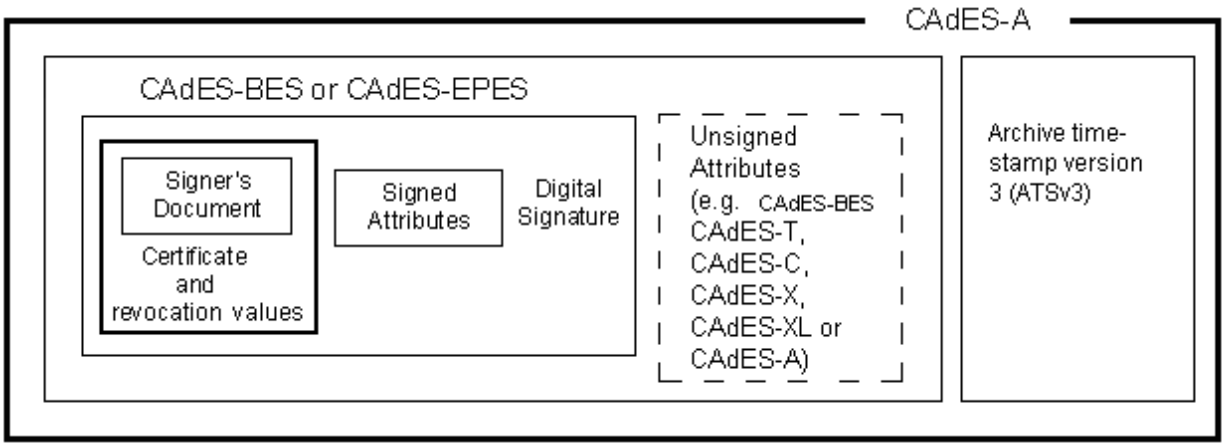


Figure 10A: Illustration of CAdES-A

4.4.5 Long-Term Electronic Signature (CAdES-LT)

Long-Term Form (CAdES-LT), in accordance with the present document builds on any format among **CAdES-T, CAdES-C, CAdES-X Long, CAdES-X Long Type 1 or 2** or a **CAdES-A** by adding one or more `long-term-validation` attributes. This form is used for archival of long-term signatures. Such an attribute can be used prior to sending the signature to a preservation service, or successive time-stamps can be used to protect the whole material against vulnerable hashing algorithms or the breaking of the cryptographic material or algorithms. Clause 6.5 contains the specification details. Clauses C.4.5 and C.4.8 provide the rationale.

NOTE: This attribute is very similar in spirit to the CAdES-X Long and CAdES-A forms. It is however simpler to implement and more flexible than these.

The use of the CAdES-LT form is deprecated for any signature to which a long-term-validation attribute has not already been applied.

An example of a structure of a CAdES-LT form is illustrated in figure 11.

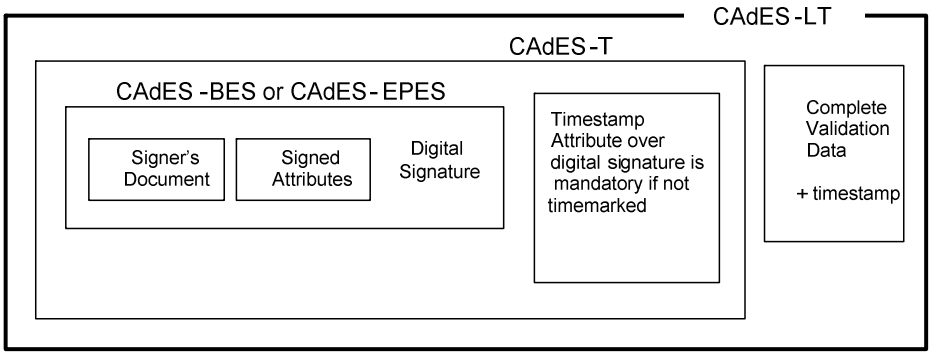


Figure 11: Illustration of CAdES-LT

4.5 Arbitration

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

- the arbitrator knows where to retrieve the signer's certificate (if not already present), all the cross-certificates and the required CRLs, ACRLs, or OCSP responses referenced in the CAdES-C;
- when time-stamping in the CAdES-T is being used, the certificate from the TSU that has issued the time-stamp token in the CAdES-T format is still within its validity period;
- when time-stamping in the 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 clause builds upon the existing Cryptographic Message Syntax (CMS), as defined in RFC 3852 [4], and Enhanced Security Services (ESS), as defined in RFC 2634 [5]. The overall structure of 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 clause 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 clause 5.7.3). The validation policy may specify requirements for the presence of certain certificates. The degenerate case, where there are no signers, is not valid in the present document.

5.5 EncapsulatedContentInfo Type

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

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

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

5.6 SignerInfo Type

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

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

The fields of type **SignerInfo** have the meanings defined in CMS (RFC 3852 [4]), but the `signedAttrs` field shall contain the following attributes:

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

5.6.1 Message Digest Calculation Process

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

5.6.2 Signature Generation Process

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

5.6.3 Signature Verification Process

The procedures for signature verification are defined in CMS (RFC 3852 [4]) and enhanced in the present document: the input to the signature verification process shall 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 [15] or SigningCertificate from ESS RFC 2634 [5] is used, the public key from the first certificate identified in the sequence of certificate identifiers from SigningCertificate shall 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]), clause 11.1.

NOTE 1: As stated in RFC 3852 [4], the content-type attribute has 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 is id-data. In both cases, the MIME content-type(s) is 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 RFC 2634 [5], shall 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 clause 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. Implementations of the present document shall support the usage of both the signing-certificate attribute and the signing-certificate-v2 attribute, within timestamp tokens, in accordance with RFC 5035 [15].

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 clause 5.8.3.

NOTE 2: Previous versions of the current document used the other signing certificate attribute (see clause 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

Earlier versions of the current document used 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 the present document 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:

- spuri: 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 contentTypes defined elsewhere; such definitions are outside the scope of the present document.

5.11 Additional Optional Attributes Defined in the Present Document

This clause defines a number of attributes that may be used to 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;

e) the mime-type of the signed document.

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

CommitmentTypeIdentifier ::= OBJECT IDENTIFIER

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

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

The following generic commitment types are defined in the present document:

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

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

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

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

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

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

These generic commitment types have the following 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 Recommendation ITU-T 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
            postalAdddress [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 clause 4.1.
```

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

NOTE 2: Attribute and AttributeCertificate are as defined respectively in Recommendations ITU-T 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 message digest as defined in clause 5.6.1 of the present document.

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

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

5.11.5 mime-type Attribute

The mime-type attribute is an attribute that lets the signature generator indicate the mime-type of the signed data. It is similar in spirit to the contentDescription field of the content-hints attribute, but can be used without a multi-layered document.

The mime-type attribute shall be a signed attribute.

The following object identifier identifies the mime-type attribute:

```
id-aa-ets-mimeType OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0) electronic-signature-standard (1733) attributes(2) 1 }
```

mime-type attribute values have ASN.1 type UTF8String:

```
mimeType::= UTF8String
```

The mimeType is used to indicate the encoding of the signed data, in accordance with the rules defined in RFC 2045 [6]; see annex F for an example of structured contents and MIME.

Only a single mime-type attribute shall be present.

The mime-type attribute shall not be used within a countersignature.

5.12 Support for Multiple Signatures

5.12.1 Independent Signatures

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

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

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 clause C.5) are supported using the countersignature unsigned attribute (see clause 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 clause specifies attributes that contain different types of validation data. These attributes build on the electronic signature specified in clause 5. This includes:

- `Signature-time-stamp` applied to the electronic signature value or a Time-Mark in an audit trail. This is defined as the Electronic Signature with Time (CAdES-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 (CAdES-C).

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

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

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

- **CAdES-X with time-stamp:** there are two types of time-stamp 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 clauses 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 clause 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 clause also specifies the data structures used in Archive validation data format (**CAdES-A)** of extended forms:

- CAdES-A form with ATSv2 attribute 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.
- CAdES-A form with ATSv3 comprises:
  - The CAdES signature to be archive time-stamped.
  - the certificates and revocation values;
  - the signed user data;
  - one or more archive-time-stamp-v3 attributes.

An archive time-stamp may be repeatedly applied after long periods to maintain validity when electronic signature and time-stamping algorithms weaken.

Finally, this clause specifies the data structures used in long-term format (**CAdES-LT)** of extended forms:

- Long-Term form of electronic signature (**CAdES-LT**) comprises:
  - The Electronic Signature with Time (CAdES-T).
  - Values of certificates and revocation information used to validate the signature, and optionally a time-stamp that applies on all that data.

A `long-term-validation` attribute may be repeatedly applied after long periods to maintain validity when electronic signature and time-stamping algorithms weaken.

The additional data required to create the forms of electronic signature identified above is carried as unsigned attributes associated with an individual signature by being placed in the `unsignedAttrs` field of `SignerInfo`. Thus, all the attributes defined in clause 6 are unsigned attributes.

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

NOTE 6: The optional attributes of the extended validation data are defined in clauses 6.3, 6.4 and 6.5.

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 clause 6.1.1 over the ES signature value.

6.1.1 signature-time-stamp Attribute Definition

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

The following object identifier identifies the `signature-time-stamp` attribute:

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

The `signature-time-stamp` attribute value has ASN.1 type `SignatureTimeStampToken:`

```
SignatureTimeStampToken ::= TimeStampToken
```

The value of 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 clause 7.4.

NOTE 1: In the case of multiple signatures, it is possible to have a:

`TimeStampToken` computed for each and all signers; or

`TimeStampToken` computed on one signer's signature; and

no `TimeStampToken` on another signer's signature.

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

6.2 Complete Validation 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 clause 6.1.1, or a time-mark operated by a **Time-Marking Authority**;
- complete-certificate-references, as defined in clause 6.2.1;
- complete-revocation-references, as defined in clause 6.2.2.

6.2.1 complete-certificate-references Attribute Definition

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

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

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

The `complete-certificate-references` attribute value has the ASN.1 syntax `CompleteCertificateRefs`.

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

`OtherCertID` is defined in clause 5.7.3.3.

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 clause 6.3.3.

This attribute may include references to the certification chain for any TSU 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

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

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

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

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

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

OcspListID ::=  SEQUENCE {
    ocspResponses        SEQUENCE OF OcspResponsesID }

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

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

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

OtherRevRefType ::= OBJECT IDENTIFIER
```

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. In previous versions of the standard, the ocspRefHash element was optional. In order to provide backward compatibility, the ASN.1 structure is not changed, however, it is strongly recommended that implementations include this element. Implementations verifying a signature may choose to accept signatures without this element, but should be warned that its absence makes OCSP responses substitutions attacks possible, if for instance OCSP responder keys are compromised. Implementations choosing to accept signatures without this element may use out-of-band mechanisms to ensure that none of the OCSP responder keys have been compromised at the time of validation.

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

NOTE 2: It is recommended that this attribute be used in preference to the `OtherRevocationInfoFormat` specified in RFC 3852 [4] to maintain backwards compatibility with the earlier version of the present document.

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 clause 5.7.3.3.

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

6.2.4 attribute-revocation-references Attribute Definition

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

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

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

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

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

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

6.3 Extended Validation Data (CAdES-X)

This clause 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 clause 6.3.5 (**CAdES-X Type 1**); or
- Time-Stamped Certificates and CRLs references, as defined in clause 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 clause 6.3.3; and
- `revocation-values` as defined in clause 6.3.4.

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

- CAdES-C Time-stamp, as defined in clause 6.3.3 (**CAdES-X long Type** 1); or
- Time-Stamped Certificates and CRLs references, as defined in clause 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 will 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 clause B.1.1 for an illustration of this form of electronic signature.

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

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

The following object identifier identifies the `certificate-values` attribute:

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

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

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

Certificate is defined in clause 7.1 (which is as defined in Recommendation ITU-T 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 clause B.1.1 for an illustration of this form of electronic signature.

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

NOTE: It is recommended that this attribute be used in preference to the OtherRevocationInfoFormat specified in RFC 3852 [4] to maintain backwards compatibility with the earlier version of the present document.

The following object identifier identifies the `revocation-values` attribute:

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

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

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

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

OtherRevValType ::= OBJECT IDENTIFIER
```

The syntax and semantics of the other revocation values (`OtherRevVals`) 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 clause 7.2 (which is as defined in Recommendation ITU-T X.509 [1]).

`BasicOCSPResponse` is defined in clause 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 clause 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.1structures 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 clause 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 clause B.1.3 for an illustration of this form of electronic signature.

A time-stamped-certs-crls-references attribute is an unsigned attribute. It is a time-stamp token issued for a list of referenced certificates and OCSP responses 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. Applications claiming conformance to the present document shall generate archive-time-stamp-v3 attributes (see clause 6.4.3) whenever a new archive time-stamp is required within the CAdES signature.

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. The archiveTimestamp attribute, defined in versions of TS 101 733 prior to 1.5.1, is **not** compatible with the attribute defined in the present document. The archiveTimestamp attribute, defined in versions 1.5.1 to 1.6.3 of TS 101 733, is compatible with the present 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 clause 7.4.

The timestamp should be created using stronger algorithms (or longer key lengths) than in the original electronic signatures and weak algorithm (key length) timestamps.

NOTE 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 shall 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.

6.4.2 ats-hash-index Attribute

The ats-hash-index attribute shall be carried as an attribute of the signature of the archive-time-stamp-v3 Attribute (see clause 6.4.3).

NOTE 1: Notation to support references to ASN.1 components is according to Recommendation ITU-T X.680 [8], clause 15.

The ats-hash-index unsigned attribute provides an unambiguous imprint of the essential components of a CAdES signature for use in the archive time-stamp (see 6.4.3). These essential components are elements of the following ASN.1 SET OF structures: unsignedAttrs, SignedData.certificates, and SignedData.crls.

NOTE 2: The SignedData.crls component as defined in RFC 3852 [4] can include OCSP and/or CRL revocation information.

The following object identifier identifies the ats-hash-index unsigned attribute:

```
  id-aa-ATSHashIndex OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0) electronic-
signature-standard(1733) attributes(2) 5 }
```

The ats-hash-index attribute value has the ASN.1 syntax ATSHashIndex:

```
    ATSHashIndex ::= SEQUENCE {
        hashIndAlgorithm AlgorithmIdentifier DEFAULT {algorithm id-sha256},
        certificatesHashIndex    SEQUENCE OF OCTET STRING,
        crlsHashIndex            SEQUENCE OF OCTET STRING,
        unsignedAttrsHashIndex   SEQUENCE OF OCTET STRING
    }
```

The field hashIndAlgorithm contains an identifier of the hash algorithm used to compute the hash values contained in certificatesHashIndex, crlsHashIndex, and unsignedAttrsHashIndex. This algorithm shall be the same as the hash algorithm used for computing the archive time-stamp's message imprint.

The field certificatesHashIndex is a sequence of octet strings. Each one contains the hash value of one instance of CertificateChoices within certificates field of the root SignedData. A hash value for every instance of CertificateChoices, as present at the time when the corresponding archive time-stamp is requested, shall be included in certificatesHashIndex. No other hash value shall be included in this field.

The field crlsHashIndex is a sequence of octet strings. Each one contains the hash value of one instance of RevocationInfoChoice within crls field of the root SignedData. A hash value for every instance of RevocationInfoChoice, as present at the time when the corresponding archive time-stamp is requested, shall be included in crlsHashIndex. No other hash values shall be included in this field.

The field unsignedAttrsHashIndex is a sequence of octet strings. Each one contains the hash value of one instance of Attribute within unsignedAttrs field of the SignerInfo. A hash value for every instance of Attribute, as present at the time when the corresponding archive time-stamp is requested, shall be included in unsignedAttrsHashIndex. No other hash values shall be included in this field.

When validating an electronic signature, the hash values of all of the certificates, revocation information and unsigned attributes are recalculated. Only those which match one of the hash values in ATSHashIndex are known to be protected by the corresponding archive time-stamp.

Each hash shall be the result, encapsulated in an octet string, of a hash computation on the entire encoded component including its tag, length and value octets. Instances of OtherCertificateFormat shall be encoded in DER, whilst preserving the encoding of any signed field included in otherCert item.

NOTE 3: Use of the ats-hash-index attribute makes it possible to add additional certificate / revocation information / unsigned attribute within SignedData.certificates / SignedData.crls / unsigned attributes of the CAdES signature (for instance counter signatures), after an archive time-stamp has been applied to a signature. Its use also allows the inclusion of components required by parallel signatures at a later time.

The ats-hash-index attribute shall have a single attribute value.

6.4.3 archive-time-stamp-v3 Attribute

The archive-time-stamp-v3 attribute is a time-stamp token of the signed document and the signature, including signed attributes, and all other essential components of the signature as protected by the ats-hash-index attribute.

The following object identifier identifies the archive-time-stamp-v3 attribute:

```
  id-aa-ets-archiveTimestampV3 OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0)
electronic-signature-standard(1733) attributes(2) 4 }
```

Attribute values of archive-time-stamp-v3 attribute have the ASN.1 syntax ArchiveTimeStampToken

```
  ArchiveTimeStampToken ::= TimeStampToken
```

The archive-time-stamp-v3 shall be an unsigned attribute. Several instances of this attribute may occur within an electronic signature both over time and from different TSUs. The archive-time-stamp-v3 attribute shall have a single attribute value.

The input for the archive-time-stamp-v3's message imprint computation shall be the concatenation (in the order shown by the list below) of the signed data hash (see bullet 2 below) and certain fields in their binary encoded form without any modification and including the tag, length and value octets:

1) The SignedData.encapContentInfo.eContentType.

2) The octets representing the hash of the signed data. The hash is computed on the same content that was used for computing the hash value that is encapsulated within the message-digest signed attribute of the CAdES signature being archive-time-stamped. The hash algorithm applied shall be the same as the hash algorithm used for computing the archive time-stamp's message imprint. The inclusion of the hash algorithm in the SignedData.digestAlgorithms set is recommended.

3) Fields version, sid, digestAlgorithm, signedAttrs, signatureAlgorithm, and signature within the SignedData.signerInfos's item corresponding to the signature being archive time-stamped, in their order of appearance.

4) A single instance of ATSHashIndex type (created as specified in clause 6.4.2).

The archive-time-stamp-v3 shall include as an unsigned attribute a single ats-hash-index, with a single ATSHashIndex AttributeValue component, added to time-stamp signature.

NOTE 1: The inclusion of the ats-hash-index unsigned attribute's component in the process that builds the input to the computation of the archive time-stamp's message imprint ensures that all the essential components of the signature (including certificates, revocation information, and unsigned attributes) are protected by the time-stamp.

The items included in the hashing procedure and the concatenation order are shown in figure 11A.

When validated, an `archive-time-stamp-v3` unsigned attribute is a proof of existence at the time indicated in its time-stamp token, of the items that have contributed to the computation of its message imprint.

NOTE 2: This proof of existence is used in validation procedures to ensure that signature validation is based on objects that truly existed in the past. This, for example, protects against a private signing key being compromised after the associated public key certificate expires resulting in the signature being considered invalid.

NOTE 3: Counter signatures held as countersignature attributes do not require independent archive time-stamps as they are protected by the archive time-stamp as an unsigned attribute.

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

Before incorporating a new `archive-time-stamp-v3` attribute, the `SignedData` shall be extended to include any validation data, not already present, which is required for validating the signature being archive time-stamped. Validation data may include certificates, CRLs, OCSP responses, as required to validate any signed object within the signature including the existing signature, time-stamps, OCSP response and certificates.

The present document specifies two strategies for the inclusion of validation data, depending on whether an ATSv2, other earlier form of archive time-stamp, as defined in previous versions of TS 101 733, has already been added to the `SignedData`:

- If none of ATSv2, an earlier form of archive time-stamp or a `long-term-validation attribute` are already present, then the new validation material shall be included within the root `SignedData.certificates`, or `SignedData.crls` as applicable.
- If an ATSv2, or other earlier form of archive time-stamp, is present then the new validation material shall be included as it is specified in clause 6.4.1 of the present document. Root `SignedData.certificates` and `SignedData.crls` contents shall not be modified.

NOTE 4: As specified in clause 6.5 once long term validation data is applied to a CAdES signature an ATSv3, or any form of archive time-stamp, cannot be used. The use of CAdES-LT in signatures to which the long term form has not already been applied is therefore deprecated.

When generating an attribute, whether initially creating the signature or extending the signature, it shall be encoded in DER, whilst preserving the encoding of any signed field included in the attribute. Extenders shall preserve the binary encoding of already present unsigned attributes and any component contributing to the archive time-stamp's message imprint computation input.

Figure 11A illustrates the hashing process:

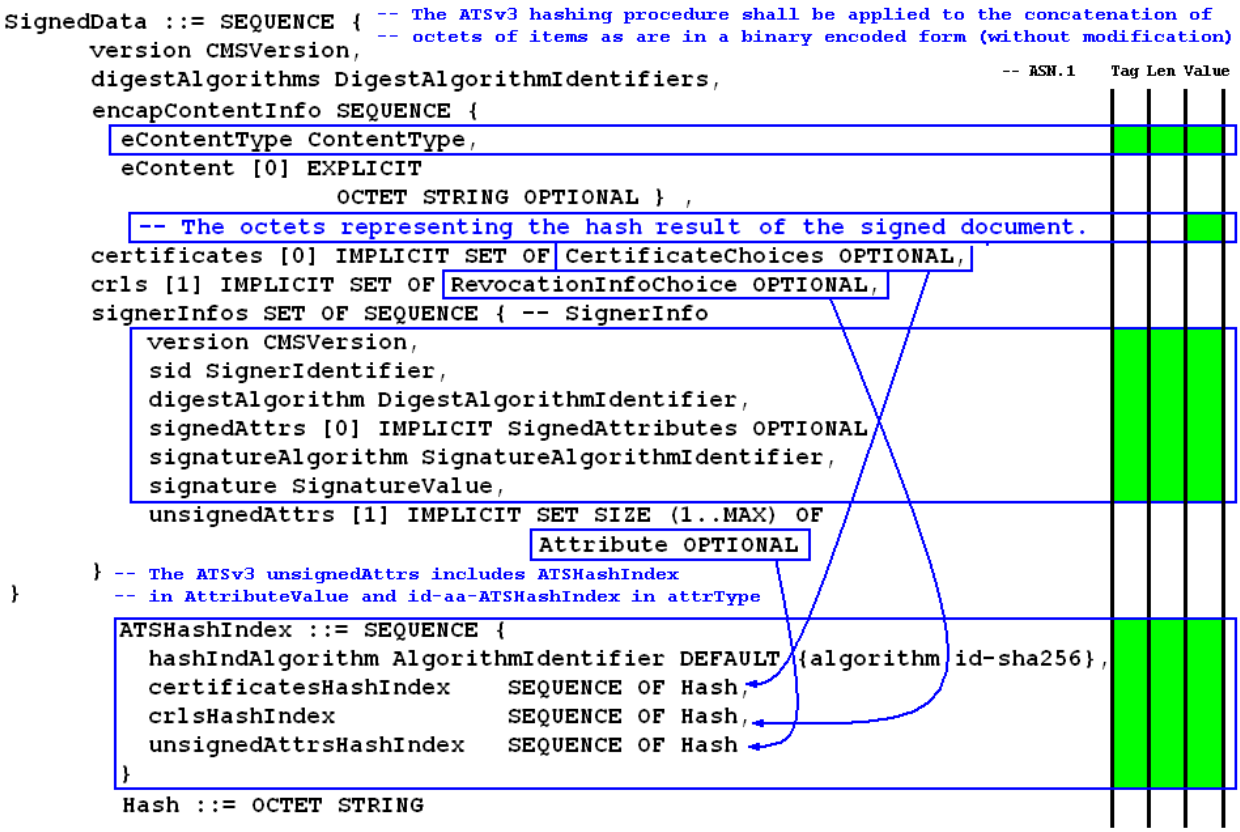


Figure 11A: Hashing process

6.5 Long-term Validation Data

6.5.1 long-term-validation Attribute Definition

The `long-term-validation` attribute is a proof of existence (PoE) at a past date, computed over many of the elements of the `SignedData` in the electronic signature. This attribute is unsigned. Several instances of this attribute may occur within the list of unsigned attributes, each instance being used in the computation of subsequently added instances. Use of this attribute is deprecated unless a long-term-validation attributes is already present in a signature.

Once a `long-term-validation` attribute has been added to an electronic signature, no other attribute may be added except other `long-term-validation` attributes, and no attribute value may be altered except possibly the last added `long-term-validation` attribute.

The following object identifier identifies the nested `long-term-validation` attribute:

```
id-aa-ets-longTermValidation OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0)
       electronic-signature-standard (1733) attributes(2) 2 }
```

Each instance of this attribute shall be encoded in its own `Attribute` structure, the `attrValues` field then consisting of a `SET` of *exactly one* value conforming to this syntax:

```
LongTermValidation ::= SEQUENCE {
        poeDate    GeneralizedTime,
        poeValue   CHOICE {
                timeStamp         [0] EXPLICIT TimeStampToken,
                evidenceRecord    [1] EXPLICIT EvidenceRecord
        } OPTIONAL,
        extraCertificates    [0] IMPLICIT CertificateSet OPTIONAL,
        extraRevocation      [1] IMPLICIT RevocationInfoChoices OPTIONAL
}
```

The poeDate records the approximate date at which this attribute is added to an electronic signature; it shall be ulterior to all poeDate values for previously added long-term-validation attributes.

The PoE itself proves the existence of some data at a precise past date, which is recorded within the PoE structure; accessing that PoE date requires decoding the PoE, a process which depends on the PoE type. The poeDate field should contain a copy of the PoE creation date; however, it is acceptable that the poeDate value is not strictly equal to the PoE creation date. This is expected in case the system which assembles the attribute instance obtains the PoE from an external system, and cannot decode it itself. In that situation, the system which adds the attribute instance should use the current date, which may be off from the PoE actual creation date by a small amount.

For long-term signature verification purposes, the poeDate field should be used only to infer the order in which the long-term-validation attribute instances were added; only the PoE date specified within the PoE itself is actually "proven". Verifiers should not fail in case the poeDate field is not equal to the PoE creation date.

The extraCertificates and extraRevocation fields are meant to receive extra validation objects which may be used to help validate the document signature, various document time-stamps or a PoE on an older long-term-validation attribute instance.

The PoE value shall be either a time-stamp as defined in RFC 3161 [7] or an evidence record as defined in RFC 4998 [16].

A time-stamp includes (in the messageImprint field of the TimeStampToken structure) a hash value computed over many of the SignedData elements. In the case of the long-term-validation attribute, a tree-like structure is used to process some elements which are, at the ASN.1 level, encoded with a SET, hence orderless.

The computation of the aforementioned hash value is done through "tree-hashing", which is a way to hash a set of data elements with a hash function *h*, such that the *ordering* of these elements is irrelevant and does not impact the final value. Tree-hashing will be used to process elements of the SignedData which are in no particular order (they are part of a SET OF) and might be subject to spurious reordering.

Given a hash function *h*, and a set of individual data elements D_i (without particular order), the elements shall be tree-hashed as follows:

- Each element D_i is hashed by itself with *h*, yielding a value V_i.
- The V_i are concatenated in ascending lexicographic order:
  - In lexicographic order, a byte sequence a_j is said to be lower than a byte sequence b_j if there exists an index k such that $a_j = b_j$ for all $j < k$, and $a_k < b_k$. Byte values are integers between 0 and 255, inclusive.
  - If there are r data elements, and the hash function outputs n bytes, then the concatenation results in a sequence of rn bytes.
  - Duplicates, if any, are not collapsed: in that case, we keep all the identical hash values in the concatenation result.
- The concatenation result is then itself hashed with *h*. That final hash output is the tree-hash result.

NOTE 1: If the tree-hash is applied on an empty sequence of data elements, the resulting output is *h(z)* where *z* is a bit string of length zero.

To obtain the hash value used as input to the PoE creation (e.g. the messageImprint value for a time-stamp), the following process shall be applied:

- A hash function *h* is used; it shall be one of the hash functions identified in the digestAlgorithms field of the SignedData.
- A hash value H_e is computed over the DER encoding of the eContentType field of the encapContentInfo of the SignedData. The Type and Length are included in the input to the hash function for the computation of H_e.

- A hash value H_m is computed over the value of the `eContent` field of the `encapContentInfo` of the `SignedData`. Only the actual data bytes are hashed, not the Type, Length, or any substructure if the `OCTET STRING` is constructed. If the `eContent` field is absent, the hash value is computed on the external protected content. H_m is the value which goes into the `message-digest` signed attribute, if the same hash function *h* is used.

- A tree-hash value H_c is computed over the elements of the `certificates` field of the `SignedData` and the extra certificates. The "data elements" for the tree-hash are the encodings (including Type and Length) of each individual `CertificateChoices` values, from both the `certificates` field, and the set of extra certificates which will be included in the new `long-term-validation` instance. The encodings as received / stored should be used.

NOTE 2: The Type and Length of the outer `SET`s are not hashed in any way. If both the `certificates` (from `SignedData`) and `extraCertificates` fields (from the to-be-added `long-term-validation` attribute) are omitted, H_c is nonetheless computed as the tree-hash of an empty set of elements (i.e. $H_c = h(z)$, the hash of the bit string of length zero).

- A tree-hash value H_r is computed over the elements of the `crls` and `extraRevocation` fields (from `SignedData` and to-be-added `long-term-validation` attribute instance, respectively) with the same method than the one used for H_c (data elements are the individual `RevocationInfoChoice` values).

- The DER-encoded `version`, `sid` and `digestAlgorithm` fields are concatenated and then hashed together, in that order, yielding H_v. The Type and Length encodings of all three fields are included (but not any Type or Length encoding from the encapsulating `SEQUENCE`).

- A hash value H_s is computed over the encoding of the signed attributes, in the same way as defined for CMS signatures (RFC 3852 [4], section 5.4): *h* is applied over the DER encoding of the `signedAttrs` field, except that a `SET OF` tag is used instead of the `IMPLICIT [0]` which is applied on that field as part of the `SignedData` structure. H_s is the input of the signature generation function which was used to sign the document in the first place, if the same hash function *h* is used.

- The DER-encoded `signatureAlgorithm` and `signature` fields are concatenated and then hashed together, in that order, yielding H_t. The Type and Length encodings of both fields are included.

- A tree-hash value H_u is computed over the unsigned attributes. Data elements are the encoded `Attribute` structures, as they are received / stored. The Type and Length of each individual `Attribute` structure are included.

- The hash values H_e, H_m, H_c, H_r, H_v, H_s, H_t and H_u are concatenated in that order. The resulting string is the input to the time-stamp generation process; in other words, the `messageImprint` field contains the value $h(H_e||H_m||H_c||H_r||H_v||H_s||H_t||H_u)$ (where "||" denotes concatenation).

Figure 12 illustrates the hashing process:

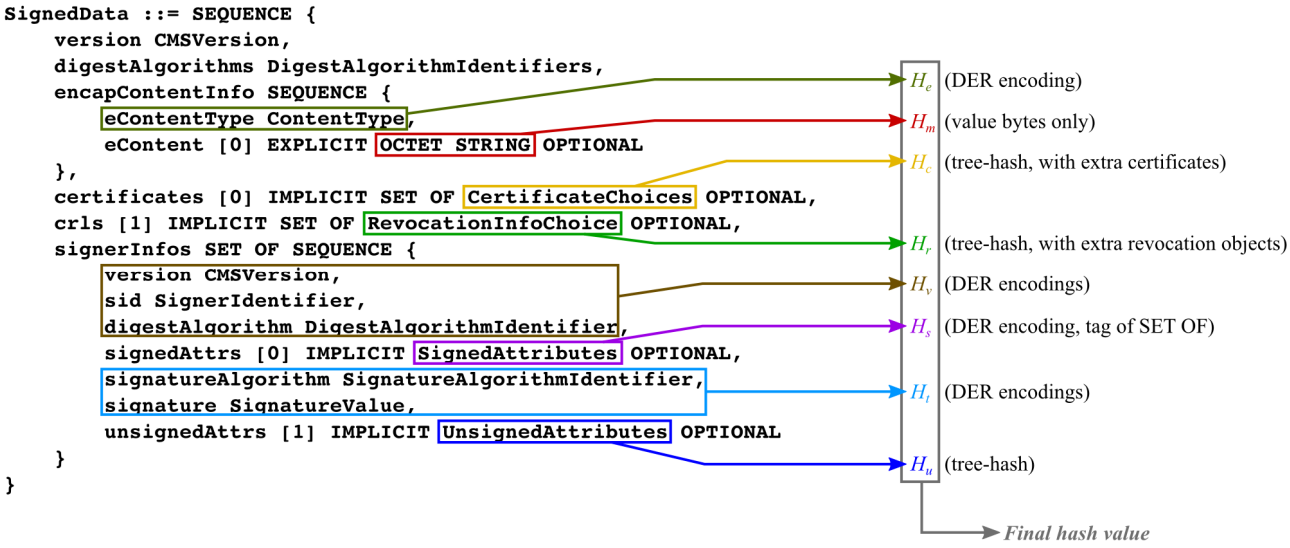


Figure 12: Hashing process

If the PoE is an evidence record, the process described above shall be done for each of the hash functions defined in the `digestAlgorithms` field of the `EvidenceRecord`. All these hash functions shall also be included in the `digestAlgorithms` field of the `SignedData`. The resulting hash values (one per used hash function *h)* are used as "`H(d*)`" in the hash tree building process described in RFC 4998 [16].

7 Other Standard Data Structures

7.1 Public Key Certificate Format

The X.509 v3 certificate basis syntax is defined in Recommendation ITU-T 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 Recommendation ITU-T 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 TS 101 861 [i.9].

Implementations of the present document shall support the usage of both the signing-certificate attribute and the signing-certificate-v2 attribute, within timestamp tokens, in accordance with RFC 5035 [15].

7.5 Name and Attribute Formats

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

NOTE 1: 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 Recommendation ITU-T 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 Recommendation ITU-T 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 2: 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 clause 7.6.

7.6 Attribute Certificate

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

7.7 Evidence Record

The format of an `EvidenceRecord` type is defined in RFC 4998 [16].

The format of `TimeStampToken` contained within these `EvidenceRecord` is defined in clause 7.4.

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 shall 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 shall be implemented.

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

On verification the inclusion of optional signed and unsigned attributes shall 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 CAdES-Basic Electronic Signature (CAdES-BES)

A system supporting CAdES-BES signers, according to the present document, shall, at a minimum, support generation of an electronic signature consisting of the following components:

- The general CMS syntax and content type, as defined in RFC 3852 [4] (see clauses 5.1 and 5.2).
- CMS SignedData, as defined in RFC 3852 [4], with the version set to either 1 or 3 as specified in section 5.1 of RFC 3852 [4] (see note 3) and at least one SignerInfo present (see clauses 5.3 to 5.6).
- The following CMS attributes, as defined in RFC 3852 [4]:
  - `content-type`; this shall always be present (see clause 5.7.1); and
  - `message-digest`; this shall always be present (see clause 5.7.2).
- One of the following attributes, as defined in the present document:
  - `signing-certificate`: as defined in clause 5.7.3.1; or
  - `signing-certificate` v2 as defined in clause 5.7.3.2.

NOTE 1: Earlier versions of the current document used the other signing-certificate attribute (see clause 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.

Implementations of the basic electronic signature form conforming to all the above requirements except the last one (e.g. without the signing-certificate or the signing-certificate v2) may use all verification attributes defined in the present document to support CMS variants of CAdES-T, CAdES-C, CAdES-X, etc. Such signatures may be referred to as CMS-T, CMS-C, CMS-X, etc.

NOTE 2: Such electronic signatures can be vulnerable to certificate substitution attacks, as described in clause C.3.3.

NOTE 3: Section 5.1 of RFC 3852 [4] requires that, the CMS SignedData version be set to 3 if certificates from SignedData is present AND (any version 1 attribute certificates are present OR any SignerInfo structures are version 3 OR eContentType from encapContentInfo is other than id-data). Otherwise, the CMS SignedData version is required to be set to 1.

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 clause 5.9:
  - `signature-policy-identifier`; this shall always be present (see clause 5.8.1).

8.3 Verification Using Time-Stamping

A system supporting verifiers, according to the present document, with time-stamping facilities shall, at a minimum, support:

- verification of the mandated components of an electronic signature, as defined in clause 8.1;
- `signature-time-stamp` attribute, as defined in clause 6.1.1;
- `complete-certificate-references` attribute, as defined in clause 6.2.1;
- `complete-revocation-references` attribute, as defined in clause 6.2.2;

- Public Key Certificates, as defined in Recommendation ITU-T X.509 [1] (see clause 8.1); and
- either of:
  - Certificate Revocation Lists, as defined in Recommendation ITU-T X.509 [1] (see clause 8.2); or
  - Online Certificate Status Protocol, as defined in RFC 2560 [3] (see clause 8.3).

8.4 Verification Using Secure Records

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

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

Annex A (normative): ASN.1 Definitions

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

A.1 Signature Format Definitions Using X.208 ASN.1 Syntax

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

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

DEFINITIONS EXPLICIT TAGS ::=
BEGIN
-- EXPORTS All

IMPORTS

-- Cryptographic Message Syntax (CMS): RFC 3852
     ContentInfo, ContentType, id-data, id-signedData, SignedData, EncapsulatedContentInfo,
     SignerInfo, id-contentType, id-messageDigest, MessageDigest, id-signingTime, SigningTime,
     id-countersignature, Countersignature
     FROM CryptographicMessageSyntax2004
     { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs-9(9)
       smime(16) modules(0) cms-2004(24) }


-- ESS Defined attributes: 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)}

-- Evidence Record Syntax RFC 4998
EvidenceRecord
    FROM ERS
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
    mechanisms(5) ltans(11) id-mod(0) id-mod-ers88(2) id-mod-ers88-v1(1)}
;


-- 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) }

id-etsi-es-attributes OBJECT IDENTIFIER ::=
    { itu-t(0) identified-organization(4) etsi(0)
       electronic-signature-standard (1733) attributes(2) }


-- 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 }


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


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

CommitmentTypeIdentifier ::= OBJECT IDENTIFIER

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


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

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

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

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

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

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


-- Signer-location attribute

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


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

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


-- 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 clause 4.1

-- Content-timestamp 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

-- Mime type

id-aa-ets-mimeType OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0)
       electronic-signature-standard (1733) attributes(2) 1}

MimeType::= UTF8String

-- Signature-timestamp attribute

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

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 OCSP response data
    producedAt          GeneralizedTime -- As in OCSP 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


-- CAdES-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

-- archive-time-stamp-v3 attribute
id-aa-ets-archiveTimestampV3 OBJECT IDENTIFIER ::= { itu-t(0)
  identified-organization(4) etsi(0)
  electronic-signature-standard(1733) attributes(2) 4 }

-- ats-hash-index attribute
id-aa-ATSHashIndex OBJECT IDENTIFIER ::= { itu-t(0)
  identified-organization(4) etsi(0)
  electronic-signature-standard(1733) attributes(2) 5 }

ATSHashIndex ::= SEQUENCE {
        hashIndAlgorithm AlgorithmIdentifier DEFAULT {algorithm id-sha256},
        certificatesHashIndex    SEQUENCE OF OCTET STRING,
        crlsHashIndex            SEQUENCE OF OCTET STRING,
        unsignedAttrsHashIndex   SEQUENCE OF OCTET STRING
}

-- Long-Term Validation attribute

id-aa-ets-longTermValidation OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0)
       electronic-signature-standard (1733) attributes(2) 2 }

LongTermValidation ::= SEQUENCE {
    poeDate    GeneralizedTime,
    poeValue   CHOICE {
        timeStamp        [0] EXPLICIT TimeStampToken,
        evidenceRecord   [1] EXPLICIT EvidenceRecord
    } OPTIONAL,
    extraCertificates   [0] IMPLICIT CertificateSet OPTIONAL,
    extraRevocation     [1] IMPLICIT RevocationInfoChoices OPTIONAL
}



-- 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 clause A.1 has precedence over that defined in clause A.2 using syntax defined in Recommendation ITU-T X.680 [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)}

-- Evidence Record Syntax RFC 4998
EvidenceRecord
    FROM ERS
    {iso(1) identified-organization(3) dod(6) internet(1) security(5)
```

```
    mechanisms(5) ltans(11) id-mod(0) id-mod-ers88(2) id-mod-ers88-v1(1)}

-- 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) }

id-etsi-es-attributes OBJECT IDENTIFIER ::=
    { itu-t(0) identified-organization(4) etsi(0)
       electronic-signature-standard (1733) attributes(2) }


-- 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 }

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


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

    PostalAddress ::= SEQUENCE SIZE(1..6) OF DirectoryString{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 clause 4.1.


-- Content Timestamp

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

ContentTimestamp:= TimeStampToken

-- Mime type

id-aa-ets-mimeType OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0)
       electronic-signature-standard (1733) attributes(2) 1}
```

```
MimeType::= UTF8String

-- Signature Timestamp

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

SignatureTimeStampToken ::= TimeStampToken

-- Complete Certificate Refs.

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

CompleteCertificateRefs ::=  SEQUENCE OF OtherCertID


-- Complete Revocation Refs

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

CompleteRevocationRefs ::=  SEQUENCE OF CrlOcspRef

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

CRLListID ::=  SEQUENCE {
    crls        SEQUENCE OF CrlValidatedID}

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

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

OcspListID ::=  SEQUENCE {
    ocspResponses        SEQUENCE OF OcspResponsesID}

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

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

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

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


-- Certificate Values

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

CertificateValues ::=  SEQUENCE OF Certificate

-- Certificate Revocation Values

id-aa-ets-revocationValues OBJECT IDENTIFIER ::= { iso(1) member-body(2)
```

```
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-aa(2) 24}

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

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

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


-- 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

-- archive-time-stamp-v3 attribute
id-aa-ets-archiveTimestampV3 OBJECT IDENTIFIER ::= { itu-t(0)
  identified-organization(4) etsi(0)
  electronic-signature-standard(1733) attributes(2) 4 }

-- ats-hash-index attribute
id-aa-ATSHashIndex OBJECT IDENTIFIER ::= { itu-t(0)
  identified-organization(4) etsi(0)
  electronic-signature-standard(1733) attributes(2) 5 }

ATSHashIndex ::= SEQUENCE {
        hashIndAlgorithm AlgorithmIdentifier DEFAULT {algorithm id-sha256},
        certificatesHashIndex    SEQUENCE OF OCTET STRING,
        crlsHashIndex            SEQUENCE OF OCTET STRING,
        unsignedAttrsHashIndex   SEQUENCE OF OCTET STRING
}

-- Long-Term Validation attribute

id-aa-ets-longTermValidation OBJECT IDENTIFIER ::= { itu-t(0) identified-organization(4) etsi(0)
       electronic-signature-standard (1733) attributes(2) 2 }

LongTermValidation ::= SEQUENCE {
    poeDate    GeneralizedTime,
    poeValue   CHOICE {
        timeStamp        [0] EXPLICIT TimeStampToken,
        evidenceRecord   [1] EXPLICIT EvidenceRecord
    } OPTIONAL,
    extraCertificates   [0] IMPLICIT CertificateSet OPTIONAL,
    extraRevocation     [1] IMPLICIT RevocationInfoChoices OPTIONAL
}-- Attribute certificate references

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

```
AttributeCertificateRefs ::=  SEQUENCE OF OtherCertID

-- Attribute revocation references

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

AttributeRevocationRefs ::=  SEQUENCE OF CrlOcspRef


END
```

Annex B (informative): Extended Forms of Electronic Signatures

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

B.1 Extended Forms of Validation Data

The Complete validation data (CAdES-C) described in clause 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 a **CAdES-X Type 2**.

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

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

A combination of the two formats above may be used. This form of eXtended validation data is called an **ES X-Long Type 1** or **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 clause C.4.6.

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

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

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

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

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

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

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

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

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

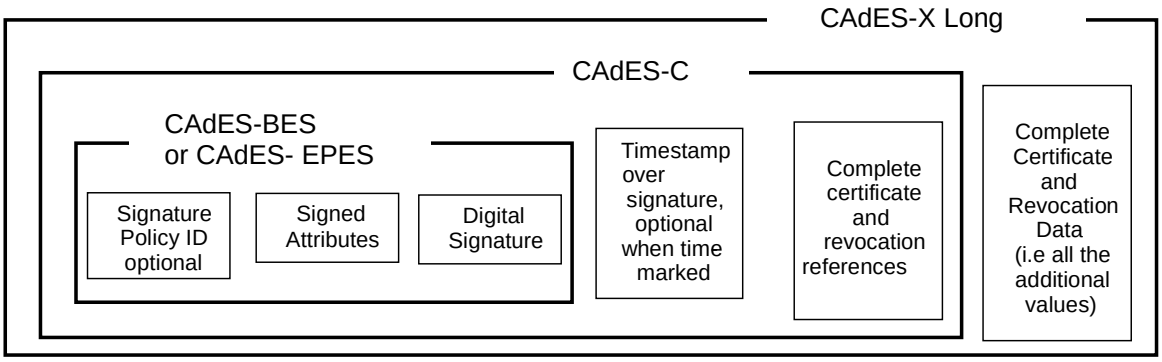


Figure B.1: Illustration of a CAdES-X long

B.1.2 CAdES-X Type 1

An electronic signature with the additional validation data forming the eXtended validation data - Type 1 X is illustrated in figure B.2 and comprises the following:

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

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

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

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

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

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

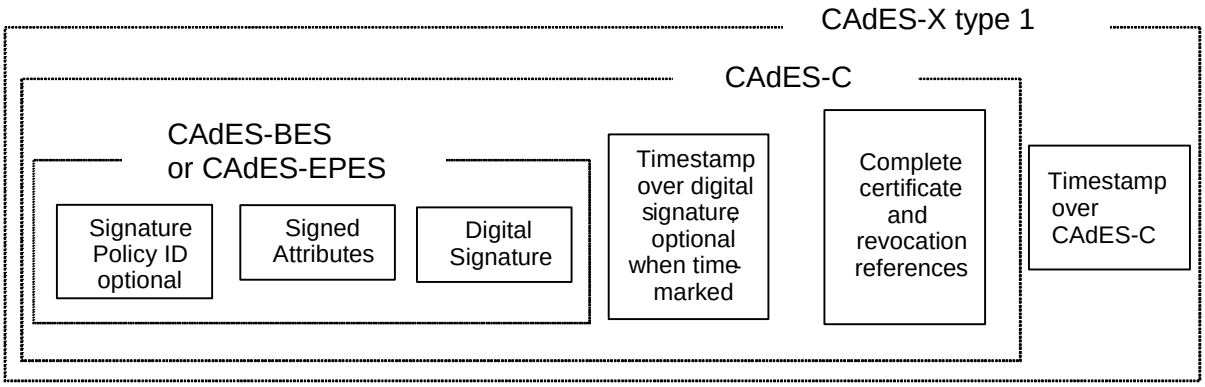


Figure B.2: Illustration of CAdES-X Type 1

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 clauses 4.2, 5.7 or 5.8;
- `complete-certificate-references` attribute, as defined in clause 6.2.1;
- `complete-revocation-references` attribute, as defined in clause 6.2.2;
- `time-stamped-certs-crls-references` attribute, as defined in clause 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 clause 6.1.1.

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

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

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

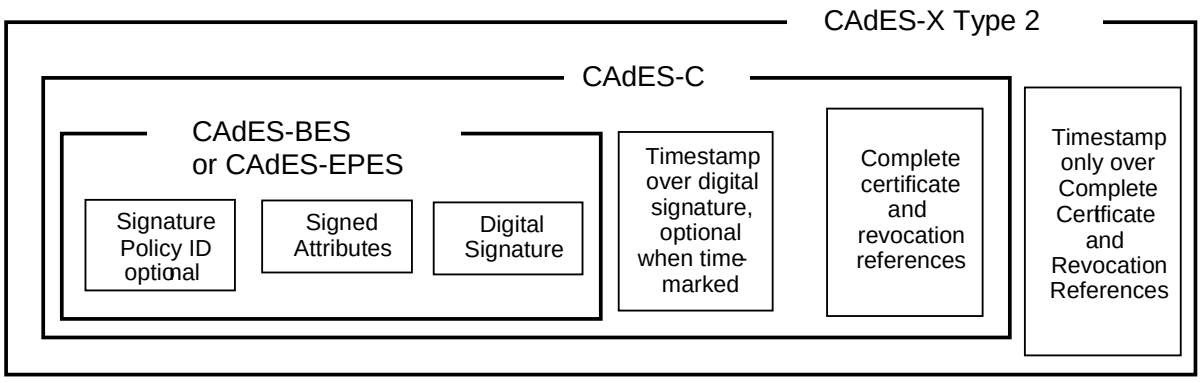


Figure B.3: Illustration of CAdES-X Type 2

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

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

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

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

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

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

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

Plus one of the following attributes is required:

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

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

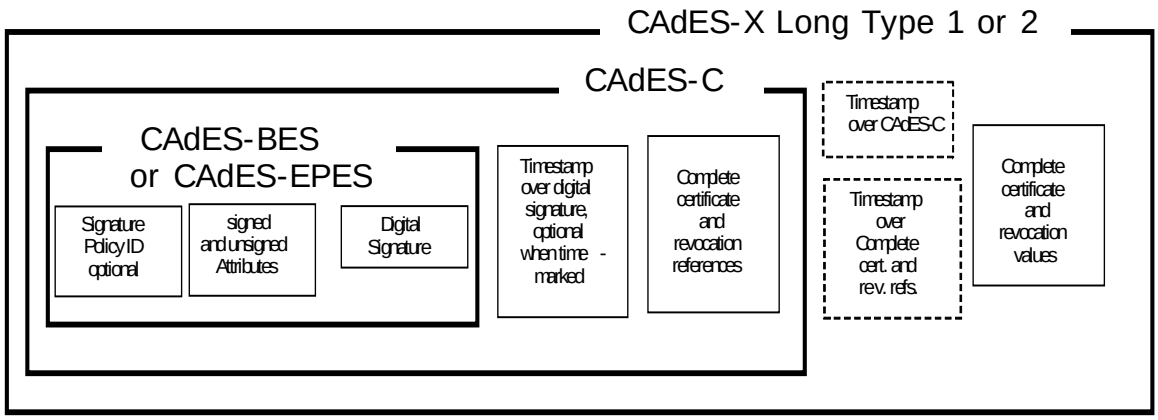


Figure B.4: Illustration of CAdES-X Long Type 1 and CAdES-X Long Type 2

B.2 Time-Stamp Extensions

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

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

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

B.3 Archive Validation Data (CAdES-A)

Before the algorithms, keys, and other cryptographic data used at the time the CAdES-C was built become weak and the cryptographic functions become vulnerable, or the certificates supporting previous time-stamps 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.

The CAdES-A comprises the following elements:

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

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

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

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

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

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

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

One of the following attributes may be present:

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

The following attribute is required:

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

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

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

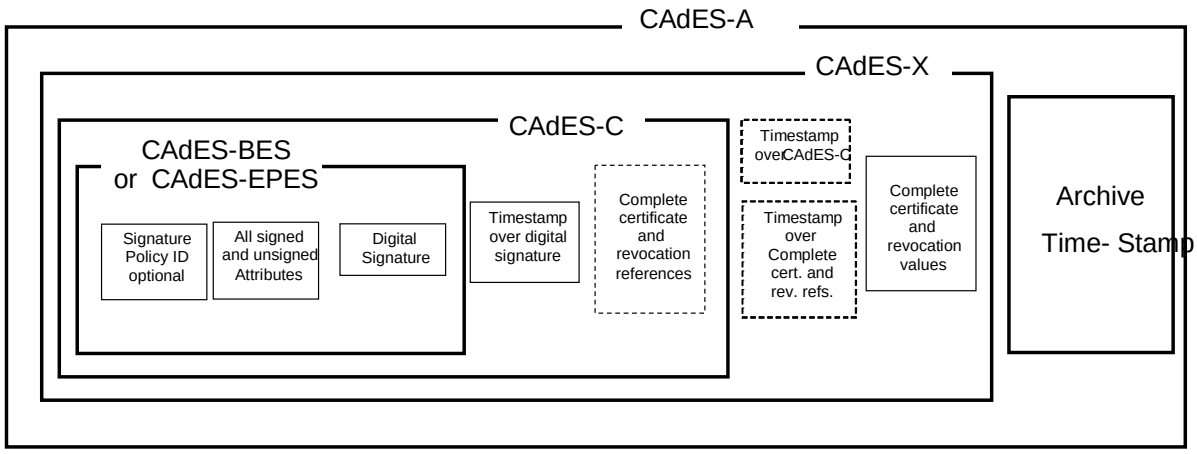


Figure B.5: Illustration of CAdES-A

B.3A Long-Term Validation Data (CAdES-LT)

As long as a validation algorithm can assess the validity of the electronic signature, the CAdES-T, or any subsequent form can be completed with a Long-Term Validation attribute.

This format is similar in spirit to the CAdES-XL and CAdES-A form. The core differences are:

- CAdES-LT can be built upon any format above CAdES-T.
- There is no restriction on the data (certificates, revocation material) that can be placed within the long-term-validation attribute.

In a CAdES-LT, it is intended that the set of certificates and revocation material be sufficient to ascertain the validation status of all end-entity certificates (signer certificate, timestamps certificates, attribute certificates, ...) contained in the electronic signature. There may be more elements than necessary and may also be less elements than necessary if it is expected that recipients have an alternate means of obtaining relevant proofs of existence on these elements.

When adding a `long-term-validation` attribute, it is expected that the electronic signature in its current state is fully verified and that all material used during validation but not already present in the signature (in places such as the `certificates` and `crls` fields of the `SignedData`, or as part of a `certificate-values` or `revocation-values` attribute) be added.

Similarly, when this attribute is added to a list of unsigned attributes which already contains one or several instances of this attribute, extra validation objects are gathered, which are needed to validate at the current date the PoE included in the most recent existing `long-term-validation` instance. "Extra objects" include the certificates and revocation objects which are not already present in that previously issued PoE.

The extra objects which are thus gathered may be included in the `extraCertificates` and `extraRevocation` fields.

An example of an electronic signature (ES) forming the CAdES-LT is illustrated in figure B.5A.

The CAdES-LT comprises the following elements:

- the CAdES-BES or CAdES-EPES, including their signed and unsigned attributes;
- a `signature-time-stamp` attribute, as defined in clause 6.1.1, if a TSP is not providing a time-mark of the ES;
- a `long-term-validation` attribute, defined in clause 6.5.1.

Several instances of the `long-term-validation` 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.

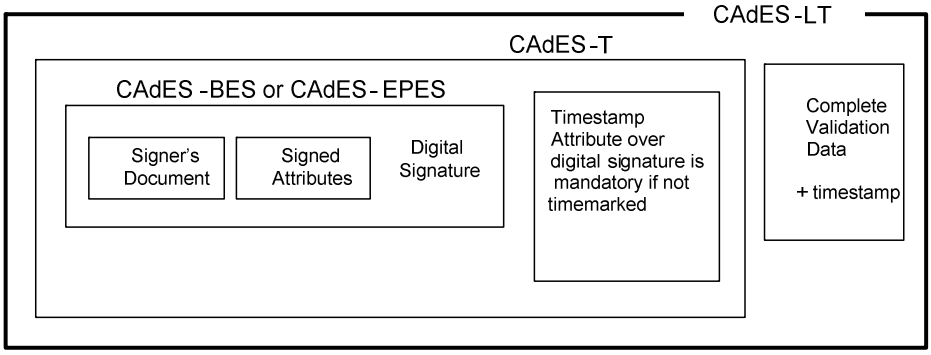


Figure B.5A: Illustration of CAdES-LT

B.3A.1 Validation

The *validation process* of a `long-term-validation` instance entails recomputing these hash values; in the case of H_u , the `long-term-validation` instance itself, and other `long-term-validation` instances which were subsequently added, should be skipped, in order to recompute the same value as the one used during the creation of the said `long-term-validation` instance.

The PoE from the `long-term-validation` instances are validated in reverse chronological order, each validation taking virtually place at the date of creation T_n of the next added PoE (which was just verified). Such a validation may use any object which has been proven to exist since at least that date T_n; this includes, in particular, the certificates and revocation objects from the `SignedData`, the older unsigned attributes, and the extra certificates and revocation objects which have been stored in the `long-term-validation` instance containing the next added PoE.

The certificates and revocation objects which are included in the `SignedData` and its attributes, and the extra objects from the `long-term-validation` instance itself, are meant to be *sufficient* to achieve validation; but they are in no way *required*: the verifier may use whatever validation process it wishes; in particular, external extra objects may be used if they can be also proven to have existed since at least the target date.

B.3A.2 Archive Update

Long-term validation uses the `long-term-validation` attributes; whenever the PoE of the last added `long-term-validation` attribute instance needs to be updated (because it will soon cease to be amenable to validation at the current date due to certificate expiration), the following process is followed:

- If the most recent `long-term-validation` instance uses an evidence record, then that `EvidenceRecord` is updated as described in RFC 4998 [16] ("timestamp renewal" or "hash-tree renewal").
- Otherwise, the `TimeStampToken` of the most recent `long-term-validation` instance is "completed" by gathering extra certificates and revocation objects needed to validate it at the current date. Then, a new `long-term-validation` instance is generated and added. The extra objects may be either pushed into the `certificates` and `crls` fields of the `TimeStampToken` from what was the most recent `long-term` instance (in which case such a push is performed *before* tree-hashing the unsigned attributes), or they could be kept separately and included in the `extraCertificates` and `extraRevocation` fields of the new `long-term-validation` instance.

NOTE: Pushing the extra certificates and revocation objects in the `certificates` and `crls` fields of the main archive `SignedData` (not that of a time-stamp) instead of storing them in the first added `long-term-validation` attribute is possible but not recommended on a general basis, because it could break the `long-term-validation` attributes of *other* `SignerInfo` structures within the same `SignedData`.

If the hash function currently used by the most recent `long-term-validation` instance is predicted to soon become "weak" (due to advances in technology or cryptanalysis), it is possible and recommended to perform the update with a new hash function. In the context of an `EvidenceRecord`, this is known as "hash-tree renewal"; but it can also be applied to the creation of a new `long-term-validation` instance. Either way, the new hash function should be added to the `digestAlgorithms` field of the `SignedData` if it is not already present.

It is acceptable to add a new `long-term-validation` instance using an evidence record, to an electronic signature whose most recent `long-term-validation` instance uses a `TimeStampToken`. The contrary (adding a time-stamp based PoE on a signature which uses an evidence record) is *not* acceptable.

B.4 Example Validation Sequence

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

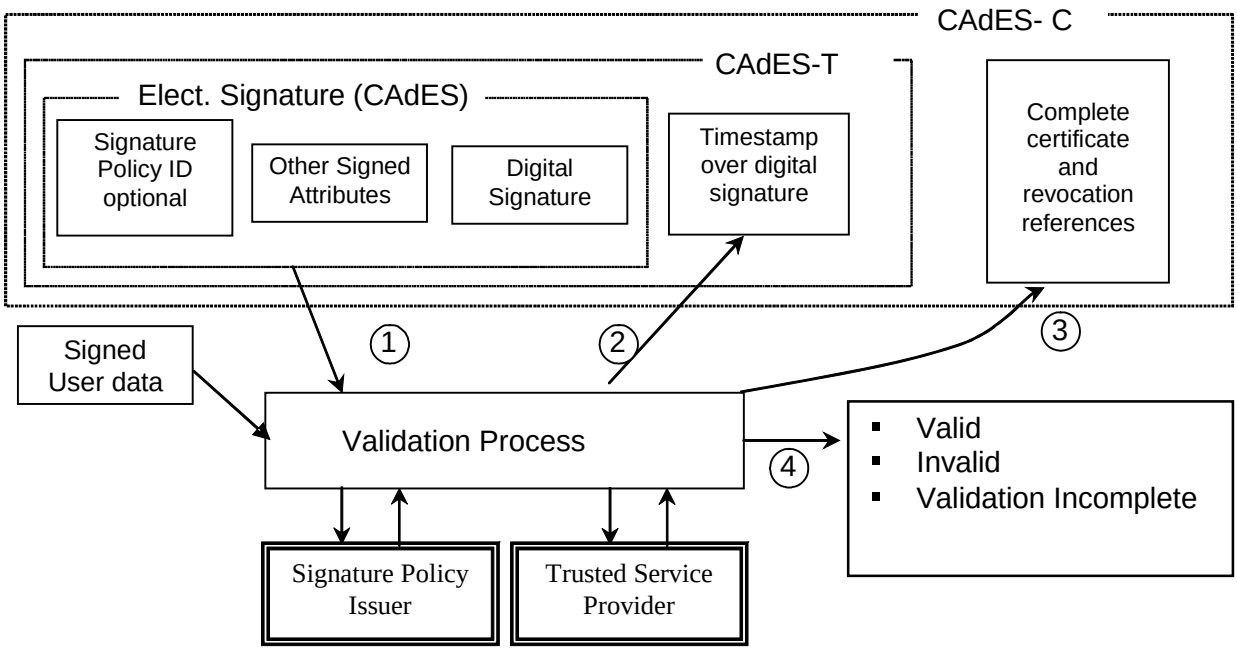


Figure B.6: Illustration of a CAdES validation sequence

Soon after receiving the electronic signature (CAdES) from the signer (1), the digital signature value may be checked; the validation process will usually at least add a time-stamp (2), unless the signer has provided one which is trusted by the verifier. The validation process may also validate the electronic signature using additional data (e.g. certificates, CRL, etc.) provided by Trusted Service Providers. When applicable, the validation process will also need to conform to the requirements specified in a signature policy. If the validation process is validation incomplete, then the output from this stage is the CAdES-T.

To ascertain the validity status as Valid or Invalid and communicate that to the user (4), all the additional data required to validate the CAdES-C needs to 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.

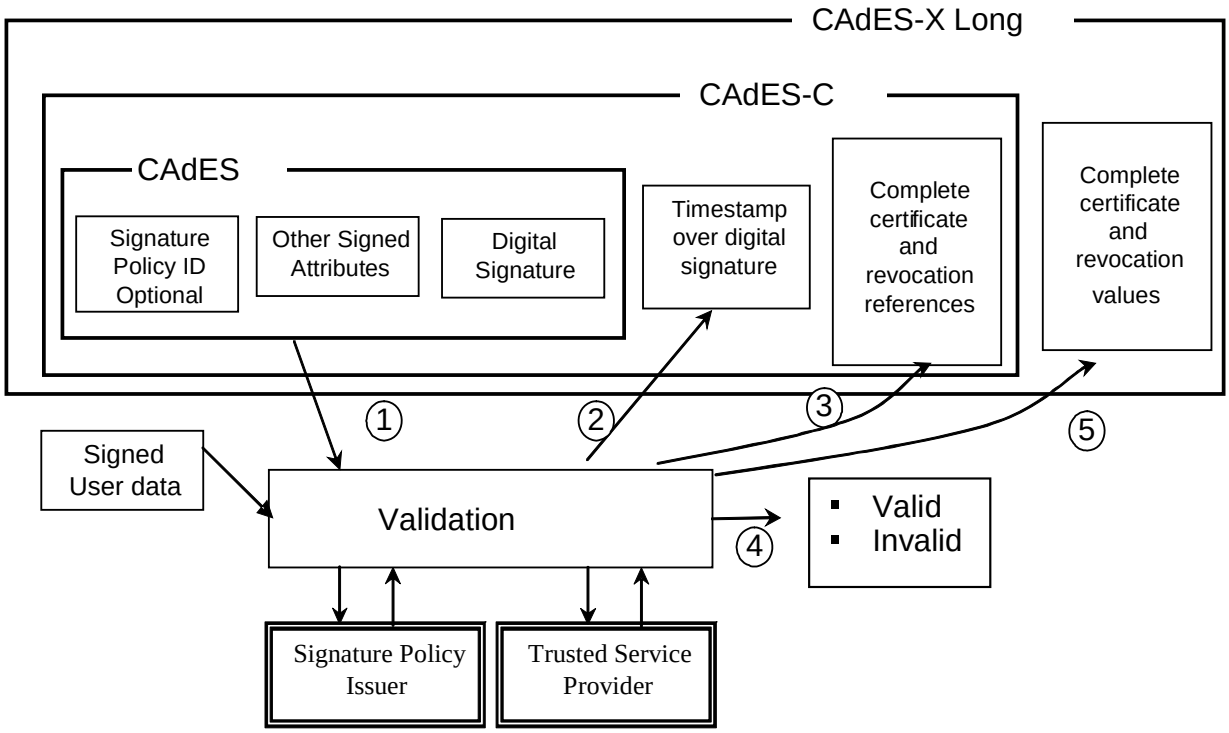


Figure B.7: Illustration of a CAdES validation sequence with CAdES-X Long

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.

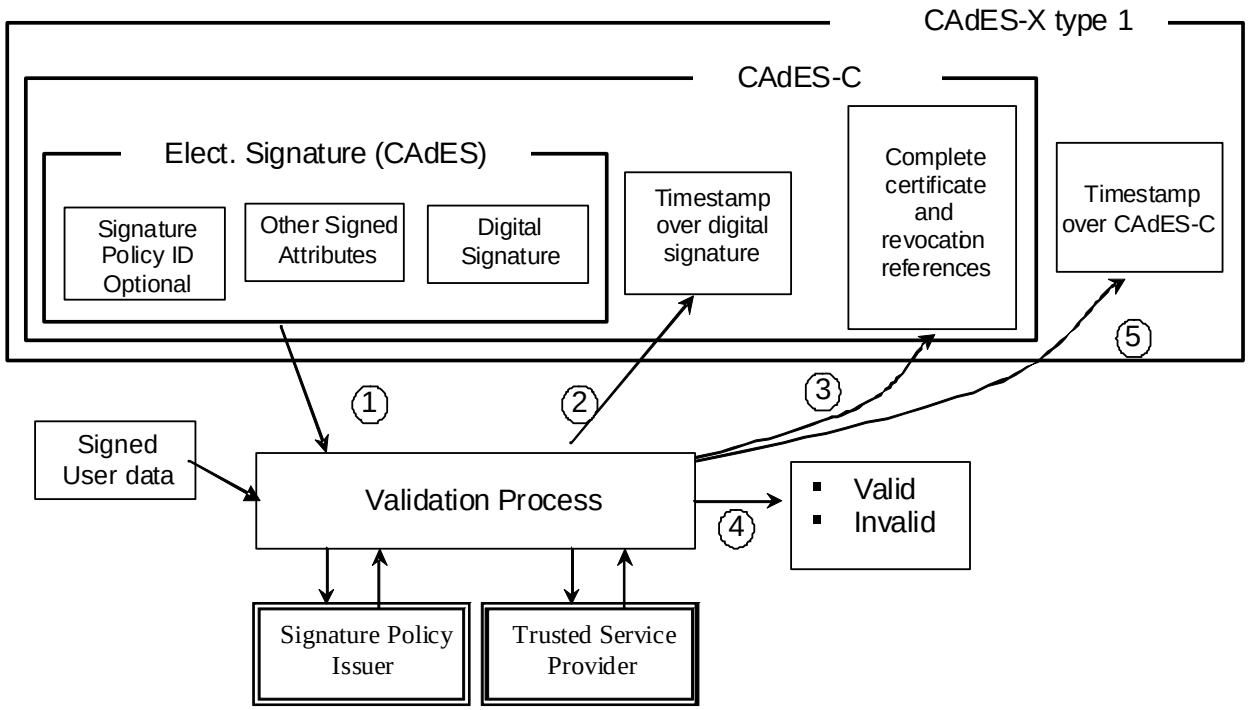


Figure B.8: Illustration of a CAdES with eXtended validation data - CAdES-X Type 1

Another alternative is to time-stamp the certificate and revocation information references used to validate the electronic signature (but not the signature) (6). The end result is called **CAdES-X Type 2**.

This is illustrated in figure B.9.

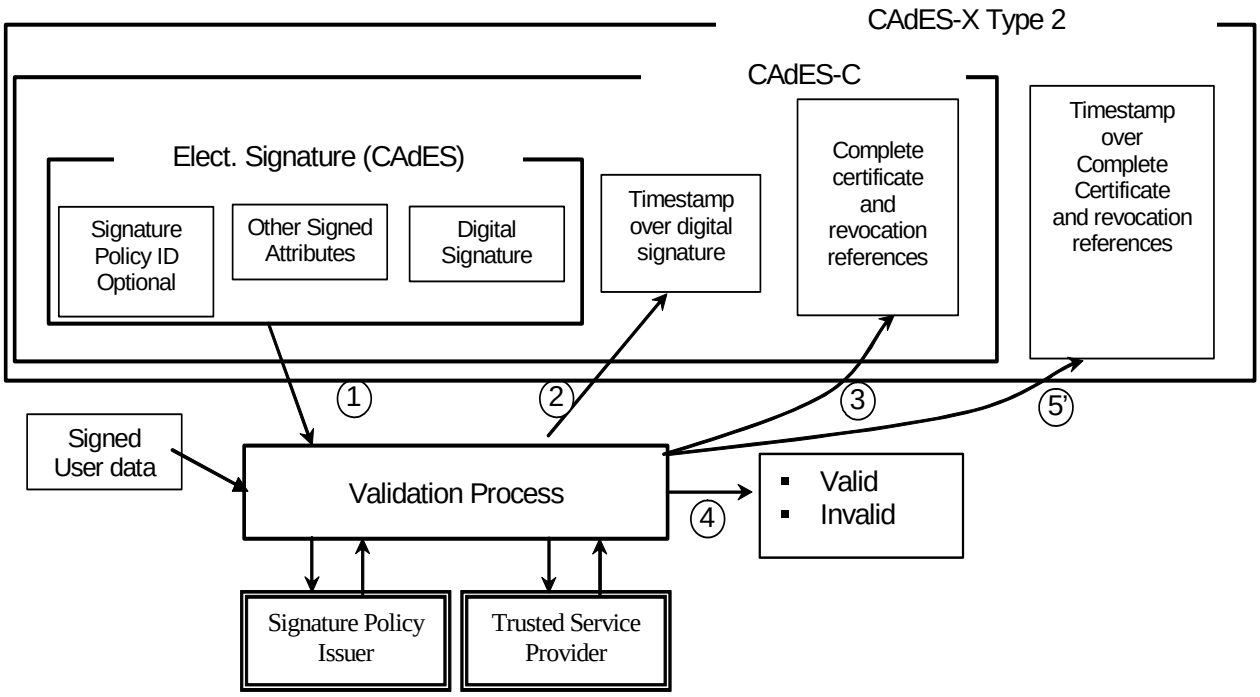


Figure B.9: Illustration of a CAdES with eXtended validation data - CAdES-X Type 2

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.

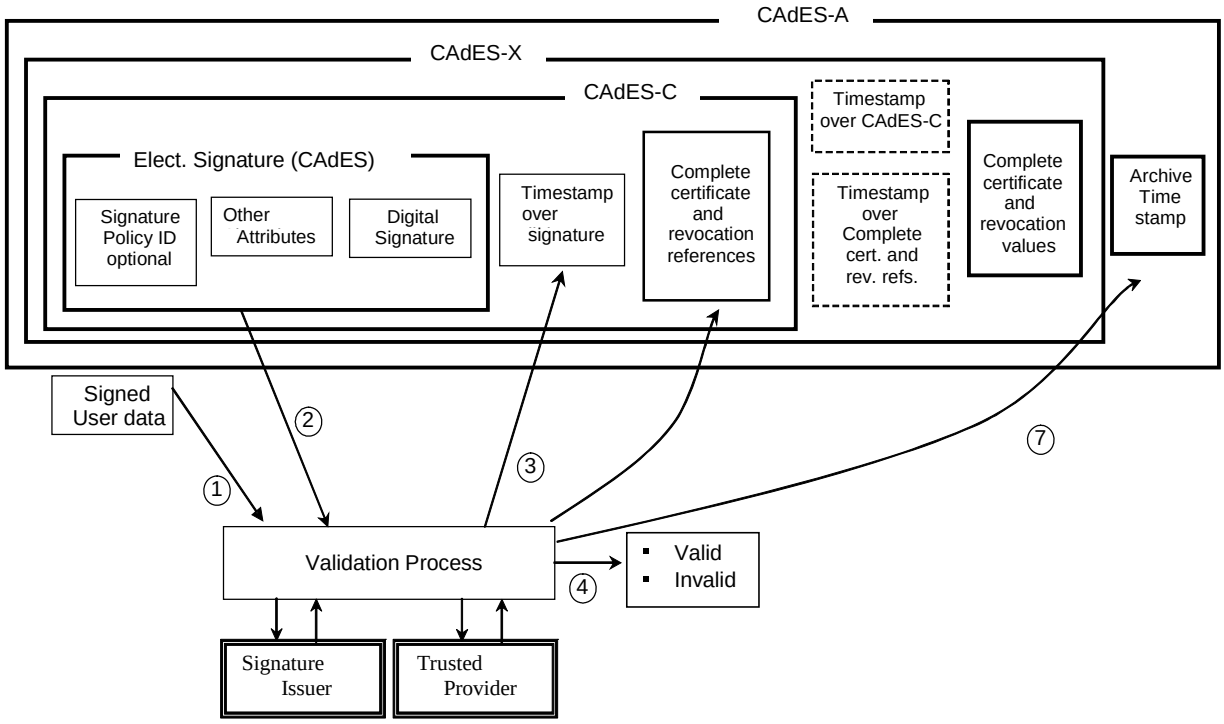


Figure B.10: Illustration of a CAdES A

B.5 Additional Optional Features

The present document also defines additional optional features to:

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

Annex C (informative): General Description

This annex explains some of the concepts and provides the 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 have to 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™ 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 to 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" is 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 is 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 is 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 is 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 can 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, he can check the two times are 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 clause 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-hints

The mime-type attribute provides information on the mime-type of the signed data.

C.3.10 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 is 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 was revoked at the time of the signature. Again, all this material may be incorporated in the electronic signature (ES X forms). An alternative would be to store 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, will 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 will 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 cryptoanalysis, or improvements in cryptoanalytical 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 2 048 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 is 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 is provided.

These forms are described in clause 5.13. All other multiple signature schemes, e.g. a signed document with a countersignature, double countersignatures, or multiple signatures, can be reduced to one or more 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 [i.10], with the schema defined in RFC 4523 [i.11].

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 [i.10], with the schema defined in RFC 4523 [i.11].

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 [i.12].

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

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"

0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAAACAAAAAgAAAAAAAAAA
EAAAtAAAAAEAAAD+////AAAAAAMAAAAGAAAA//////////////////////////////////////
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 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 RFC 3851 [i.13]).

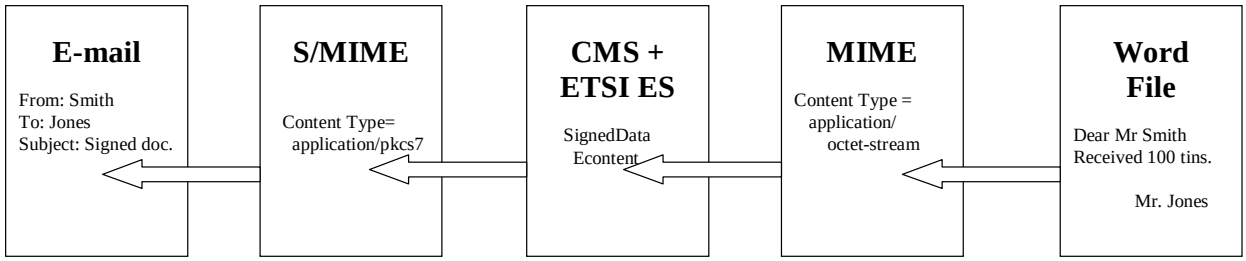


Figure F.1: Illustration of relation of using S/MIME

S/MIME carries electronic signatures as either:

- an "application/pkcs7-mime" object with the CMS carried as binary attachment (PKCS7 is the name of the early version of CMS).
  - The signed data may be included in the SignedData, which itself may be included in a single S/MIME object. See RFC 3851 [i.13], clause 3.4.2: "Signing Using application/pkcs7-mime with SignedData" and figure F.2.

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 RFC 3851 [i.13], clause 3.4.3: "Signing Using the multipart/signed Format" and figure F.3.

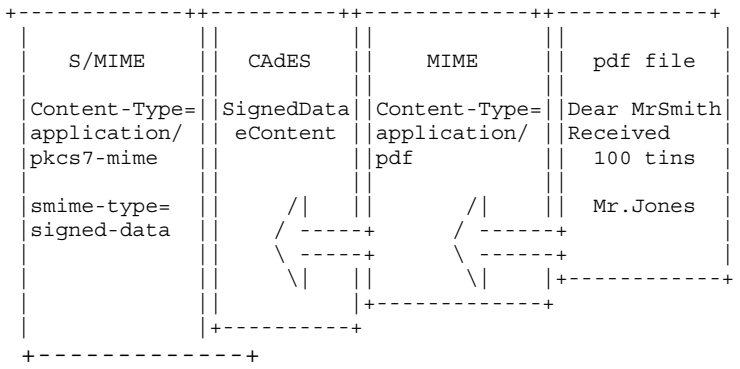


Figure F.2: Signing Using application/pkcs7-mime

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
  HUujhJh4VQpfyF467GhIGfHfYGTrfvbnjT6jH7756tbB9H7n8HHGghyHh
  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 RFC 3851 [i.13], clause 3.4.3: "Signing Using the multipart/signed Format" and figure F.3 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.3. The CMS structure just holds the electronic signature.

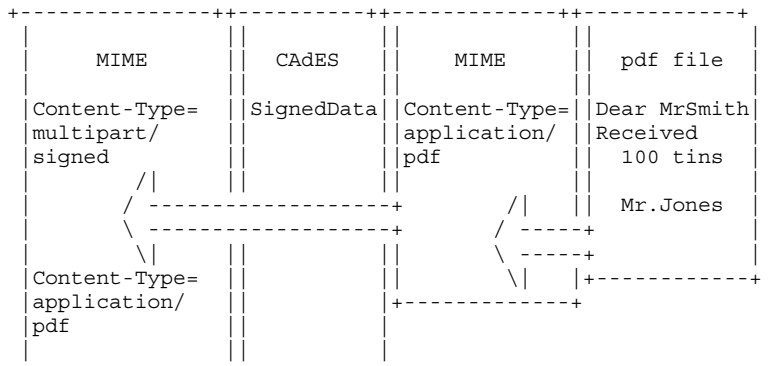


Figure F.3: Signing Using application/pkcs7-signature

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 [i.5].

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 [i.5].

G.2 Electronic Signatures and the Directive

This Directive [i.5] 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 [i.5] 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 [i.5] 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 [i.5] and is issued by a "certification-service-provider" that meets requirements given in annex II of the Directive [i.5]. 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 [i.5]. 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 [i.5];

b) considered to be an "advanced electronic signature" under the terms of the Directive [i.5];

c) considered to be a "Qualified Electronic Signature", provided the additional requirements in annexes I, II, and III of the Directive [i.5] are met. The requirements in annexes I, II, and III of the Directive [i.5] 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 [i.5]. 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 [i.5]. 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 [i.5].

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 clause 5 and conformance clause 10.1) is applicable to the proposed class of electronic signature: "qualified electronic signatures fulfilling article 5.1".

With the addition of attributes by the verifier (see clause 6 and conformance clause 10.2) the qualified electronic signature supports the 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 Object Management Group (OMG).

H.1 Data Framing

In order to be able to use either of these APIs, it will be necessary to frame the previously defined electronic signature data structures using a mechanism-independent token format. Clause 3.1 of RFC 2743 [i.14] 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 are 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:

5) 0x60 -- Tag for RFC 2743 [i.14] SEQUENCE; indicates that constructed form, definite length encoding follows.

6) 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:

   - If the indicated value is less than 128, it is represented in a single octet with bit 8 (high order) set to "0" and the remaining bits representing the value.

   - If the indicated value is 128 or more, it is represented in two or more octets, with bit 8 of the first octet set to "1" and the remaining bits of the first octet specifying the number of additional octets. The subsequent octets carry the value, 8 bits per octet, most significant digit first. The minimum number of octets is used to encode the length (i.e. no octets representing leading zeros are included within the length encoding).

7) 0x06 -- Tag for OBJECT IDENTIFIER.

8) 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.

9) 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 is used to encode each component (i.e. no octets representing leading zeros are 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 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 Common Authentication Technology working group produced in December 1998, an RFC (RFC 2479 [i.15]) 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.

Clause 2.3.3 of IDUP [i.15] describes the specific calls needed to handle evidence. This 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** (see RFC 2479 [i.15]) 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 (see RFC 2479 [i.15]), 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 180-2 [i.39]. 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 contains 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 [i.16] contains definitions and describes basic concepts.
- ISO/IEC 10118-2 [i.17] specifies two ways to construct a hash-function from a block cipher.
- ISO/IEC 10118-3 [i.18] specifies the following dedicated hash-functions:
  - SHA-1 (FIPS PUB 180-1 [i.42]);
  - RIPEMD-128;
  - RIPEMD-160.
- ISO/IEC 10118-4 [i.19].
- FIPS PUB 180-2 [i.39]: four secure hash algorithms - SHA-1, SHA-256, SHA-384, and SHA-512. The SHA-1 algorithm was first published in 1993, was slightly revised in 1995 and renamed SHA-1 in FIPS PUB 180-1 [i.42].
- ANSI X9.30-2 [i.20] specifies the ANSI-Version of SHA-1.
- ANSI X9.31-2 [i.21] specifies hash algorithms.

I.2 Digital Signature Algorithms

I.2.1 DSA

The DSA signature algorithm is defined in FIPS PUB 186-2 [i.43]. 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 is not present.

I.2.2 RSA

The RSA signature algorithm is defined in RFC 3447 [i.22]. RFC 3370 [10] specifies the use of the RSA signature algorithm with the SHA-1 algorithm. The algorithm identifier for RSA with SHA-1 is:

```
Sha1WithRSAEncryption OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) rsadsi(113549)
   pkcs(1) pkcs-1(1) 5 }
```

NOTE: RFC 3370 [10] recommends that MD5 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 PUB 186-2 [i.43]. NIST's *Digital Signature Algorithm* (DSA) is a variant of ElGamal Discrete Logarithm-based digital signature mechanism. The DSA requires a 160-bit hash-function and mandates SHA-1.
- IEEE 1363 [i.23] 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, ECDSA, 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 [i.24] specifies a digital signature mechanism based on the RSA public-key technique and a specifically designed redundancy function.
- ISO/IEC 9796-2 [i.25] 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 9796-3 [i.26] 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 [i.27] contains definitions and describes the basic concepts of digital signatures with appendix.
- ISO/IEC 14888-2 [i.28] 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 [i.29] specifies digital signature schemes with appendix that make use of certificate-based keying material. The document includes the following schemes:
  - DSA;
  - ECDSA, an elliptic curve-based analog of NIST's Digital Signature Algorithm;
  - Pointcheval-Vaudeney signatures;
  - RSA signatures;
  - Identity based signature schemes on an additive group of elliptic curve points.
- ISO/IEC 15946-2 [i.37].
- ISO/IEC 11770-3 [i.30] specifies digital signature schemes with appendix using elliptic curves. The document includes two schemes:
  - ECDSA, 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.TR 31 [i.40]. ANSI X9.31-1 [i.41] specifies a digital signature mechanism with appendix using the RSA public key technique.
- ANSI X9.30.1 [i.31] specifies the DSA, NIST's *Digital Signature Algorithm*.
- ANSI X9.62 [i.32] specifies the *Elliptic Curve Digital Signature Algorithm*, an analog of NIST's *Digital Signature Algorithm* (DSA) using elliptic curves. The appendices provide tutorial information on the underlying mathematics for elliptic curve cryptography and many examples.

Annex J (informative): Guidance on Naming

J.1 Allocation of Names

The subject name is 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 verifies 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 2616 [i.33] (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 [i.34]) 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.

Annex K (informative): Time-stamp hash calculation

Tables K.1 to K.3 identify the components of the signatures to be used as input for the computation of the message imprint of the time-stamp.

Table K.1: The identification of attributes from which the hash is calculated

Type of Timestamp	Identification
id-aa-ets-contentTimestamp	D
id-aa-signatureTimeStampToken	S
id-aa-ets-escTimeStamp	C
id-aa-ets-certCRLTimestamp	R
id-aa-ets-archiveTimestampv2	A
id-aa-ets-archiveTimestampv3	A3

ASN.1 data elements which are already included in a hash within the value of the containing (ASN.1) object are identified by a "**".

Table K.2: CAdES Signature

ASN.1
ContentInfo ::= SEQUENCE {
contentType ContentType, -- id-signedData
content [0] EXPLICIT ANY DEFINED BY contentType }

Table K.3: SignedData

	ASN.1	Tag	Len	Value
1	SignedData ::= SEQUENCE { version CMSVersion, digestAlgorithms DigestAlgorithmIdentifiers,			
2	encapContentInfo SEQUENCE {	A	A	A
3	eContentType ContentType,	**, A3	**, A3	**, A3
4	eContent [0] EXPLICIT	**	**	**
5	OCTET STRING OPTIONAL	**	**	**, D(2)
6	-- External Data if external signatures (detached)			A(1), D(1)
7	-- The octets representing the hash value -- of the signed data },			A3
8	certificates [0] IMPLICIT SET OF	A	A	A
9	CertificateChoices OPTIONAL,	**	**	**
10	crls [1] IMPLICIT SET OF	A	A	A
11	RevocationInfoChoice OPTIONAL,	**	**	**
12	signerInfos SET OF SEQUENCE { -- SignerInfo parallel signatures			
13	version CMSVersion,	A, A3	A, A3	A, A3
14	sid SignerIdentifier,	A, A3	A, A3	A, A3
15	digestAlgorithm DigestAlgorithmIdentifier,	A, A3	A, A3	A, A3
16	signedAttrs [0] IMPLICIT SET SIZE (1.. MAX) OF	A, A3	A, A3	A, A3
17	SEQUENCE{ -- Attribute	**	**	**
18	attrType OBJECT IDENTIFIER,	**	**	**
19	attrValues SET OF AttributeValue } OPTIONAL,	**	**	**
20	signatureAlgorithm SignatureAlgorithmIdentifier,	A, A3	A, A3	A, A3

	ASN.1	Tag	Len	Value
21	signature OCTET STRING – SignatureValue,	A, A3	A, A3	S, C, A, A3
22	unsignedAttrs [1] IMPLICIT SET SIZE (1..MAX) OF	$A^{(3,4)}$	$A^{(3,4)}$	$A^{(3,4)}$

23	SEQUENCE { -- Attribute
	
	-- if attrType is id-aa-signatureTimeStampToken
	-- if attrType is id-aa-ets-certificateRefs
	-- if attrType is id-aa-ets-revocationRefs
	-- Attribute ::= SEQUENCE {
	-- attrType OBJECT IDENTIFIER,
	-- attrValues SET OF AttributeValue
	} OPTIONAL
	}
	}
		**	**	**
				
				C
				C, R
				C, R

24	-- The ATSv3 incorporates ATSHashIndex as an -- unsigned attribute, which also contributes to -- its message imprint computation. ATSHashIndex ::= SEQUENCE {	A3	A3	A3
25	hashIndAlgorithm AlgorithmIdentifier DEFAULT {algorithm id-sha256},	**	**	**
26	certificatesHashIndex SEQUENCE OF OCTET STRING,	**	**	**
27	crlsHashIndex SEQUENCE OF OCTET STRING,	**	**	**
28	unsignedAttrsHashIndex SEQUENCE OF OCTET STRING	**	**	**
29	}			

NOTE 1: External data means data protected by a detached signature that is not included in the CAdES signature eContent. The hash computation for contentTimestamp and archiveTimestamp includes hash computation through the external data. Algorithms used in the detached signature may become weak in the future, that is why the integrity of external data needs to be also protected with archiveTimestamp.

NOTE 2: There is a small difference in treatment between the content-time-stamp and the archive-timestamp (ATSv2) when the signature is attached. In that case, the content-timestamp is computed on the raw data (without ASN.1 tag and length) whereas the archive-timestamp is computed on the data as read.

NOTE 3: A previous version of CAdES did not include the tag and length octets of this SET OF type of unsignedAttrs element in this annex, which contradicted the normative section. To maximize interoperability, it is recommended to simultaneously compute the two hash values (including and not including the tag and length octets of SET OF type) and to test the value of the timestamp against both.

NOTE 4: It is difficult to achieve interoperability between implementations using purely DER and implementations using purely BER when multiple archive time-stamps are present, since the order of attribute hashing may be hard to rebuild in these situations. The attributes defined in clause 6.4.3 or defined in clause 6.5 can be used regardless of the encoding/reencoding methods used by different implementations.

Annex L (informative): Changes from the previous version

L.1 Changes before 1.7.4

The major changes are as follows :

- the title of the document has changed to be aligned with the title of XAdES;
- the vocabulary used within the present document has been aligned with the vocabulary used in XAdES;
- the OIDs from the ASN.1 modules have changed for the following reasons:
  - the OIDs of the ASN.1 modules of RFC 2560 [3] and RFC 3161 [7] have been included;
  - since RFC 5280 [i.35] and RFC 3369 [i.36] have been obsoleted by RFC 3280 [2] and RFC 3852 [4] respectively, there was the need to refer to the OIDs of the ASN.1 modules of RFC 3280 [2] and RFC 3852 [4], instead of the OIDs of the ASN.1 modules of RFC 5280 [i.35] and RFC 3369 [i.36];
- if the hash of the signature policy is unknown, then, by convention, the sigPolicyHash is set to all zeros;
- the Use of ESS Signing Certificate V2 is added for use of hash algorithms other than SHA-1 in hashing certificates instead of "other-signing-certificate";
- archive timestamp OID changed to avoid problems of backward compatibility and processing clarified;
- numerous editorial changes to align with IETF equivalent RFC specification.

L.2 Changes between 1.7.4 and 1.8.1

The major changes are as follows:

- Clarification on the way to compute the content-timestamp attribute.
- Added the ability to add hash algorithm agility to timestamps.
- Provided guidance on the usage of the complete-certificate-references attributes with OCSP.
- Provided guidance on the usage of attributes defined in the present document with "bare" CMS.
- Added annex on timestamp hash computations.
- Correction of clause B.3 to make time stamps optional for CAdES-A.

L.3 Changes between 1.8.1 and 1.8.3

- Clarification on conformance requirement for CAdES-BES signatures.
- Correction clause A.1. "SignaturePolicy" corrected to "SignaturePolicyIdentifier".

L.4 Changes between 1.8.3 and 2.1.1

- Clarification on the informative computation of the archive timestamp.
- Addition of a long-term-validation attribute.
- Addition of a mime-type attribute.

L.5 Changes between 2.1.1 and 2.2.1

- Addition of new archive time-stamp format *archive-time-stamp-v3* (ATSv3) and associated *ats-hash-index.*
- Use of long-term-validation attribute deprecated and new ATSv3 required for new archive time-stamps.
- Changes to Annex K time-stamp hash calculation.

Annex M (informative): Bibliography

- IETF RFC 2246 (1999): "The TLS Protocol Version 1.0".
- ETSI TS 102 023: "Electronic Signatures and Infrastructures (ESI); Policy requirements for time-stamping authorities".
- XMLDSIG: W3C/IETF Recommendation (February 2002): "XML-Signature Syntax and Processing".
- Recommendation ITU-T X.690 (2002): "Information technology ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

History

Document history		
V1.1.3	May 2000	Publication as ES 201 733
V1.2.2	December 2000	Publication
V1.3.1	February 2002	Publication
V1.4.0	September 2002	Publication
V1.5.1	December 2003	Publication
V1.6.3	September 2005	Publication
V1.7.3	January 2007	Publication
V1.7.4	July 2008	Publication
V1.8.1	November 2009	Publication
V1.8.3	January 2011	Publication
V2.1.1	March 2012	Publication
V2.2.1	April 2013	Publication