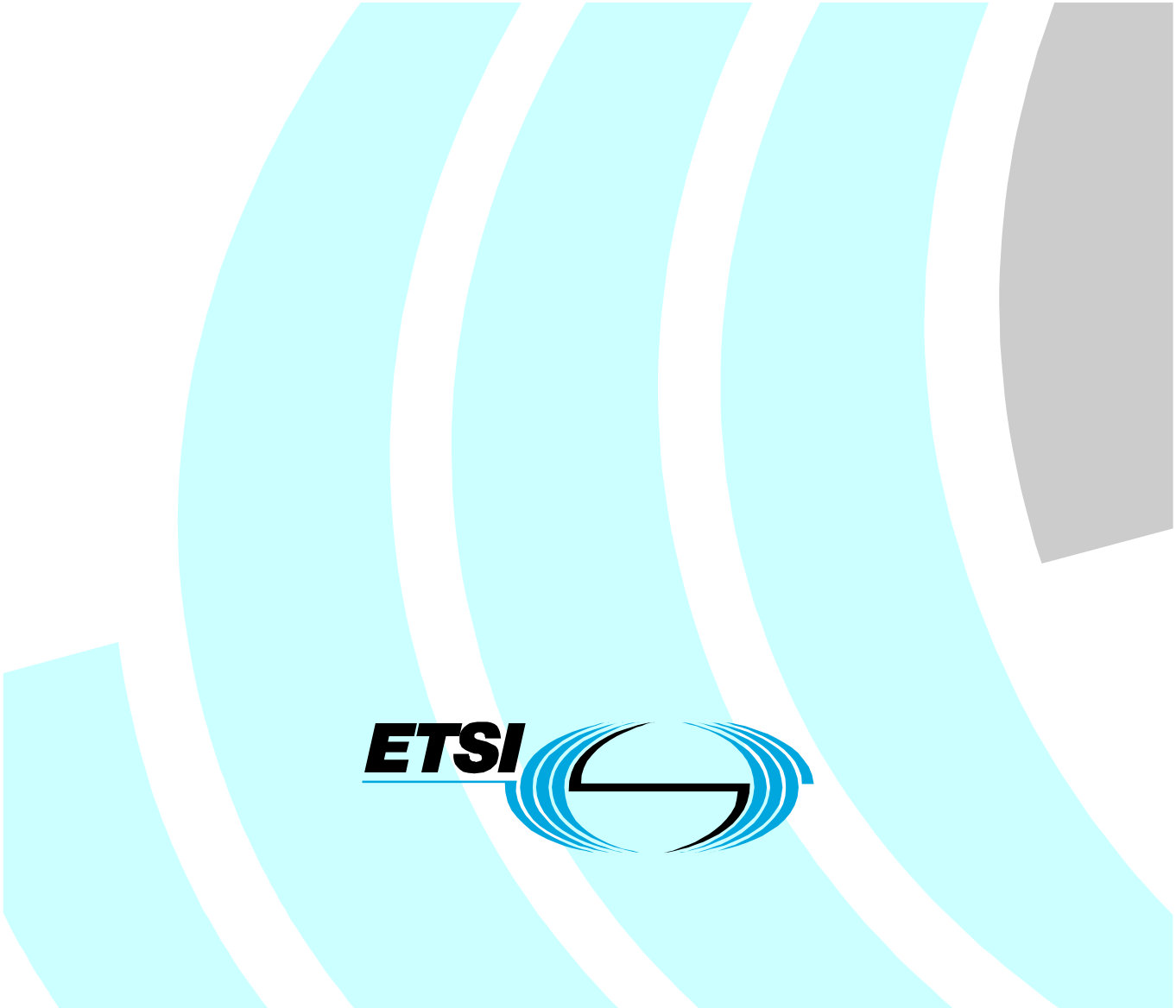
ETSI TR 102 272 V1.1.1 (2003-12)

Technical Report

Electronic Signatures and Infrastructures (ESI); ASN.1 format for signature policies



Reference
DTR/ESI-000022

Keywords
ASN.1, IP, electronic signature, security,
e-commerce

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 5
Foreword 5
Introduction 5
1 Scope 6
2 References 6
3 Definitions and abbreviations 7
3.1 Definitions 7
3.2 Abbreviations 8
4 Signature Policy overview 8
5 Signature policy specification in informal free text form 9
6 Signature policy specification in ASN.1 12
6.1 Overall ASN.1 structure 13
6.2 Signature validation policy 13
6.3 Common Rules 14
6.4 Commitment Rules 14
6.5 Signer and Verifier Rules 15
6.5.1 Signer rules 15
6.5.2 Verifier rules 16
6.6 Certificate and revocation requirement 16
6.6.1 Certificate requirements 16
6.6.2 Revocation requirements 17
6.7 Signing certificate trust conditions 18
6.8 Time-Stamp trust conditions 18
6.9 Attribute trust conditions 19
6.10 Algorithm constraints 19
6.11 Signature policy extensions 20
Annex A: ASN.1 modules 21
A.1 Signature policies definitions using X.208 (1988) ASN.1 syntax 21
A.2 Signature policy definitions using X.680 (2002) ASN.1 syntax 24
Annex B: What is a signature policy and signature validation policy 29
B.0 Introduction 29
B.1 Identification of signature policy 30
B.2 General signature policy information 31
B.3 Recognized commitment types 31
B.4 Rules for use of certification authorities 32
B.4.1 Trust points 32
B.4.2 Certification path 32
B.5 Rules for the use of time-stamping and time-marking 33
B.5.1 Trust points and certificate paths 33
B.5.2 Time-stamping authority names 33
B.5.3 Timing constraints - cautionary period 33
B.5.4 Timing constraints - time-stamp delay 33
B.6 Revocation rules 34
B.7 Rules for the use of roles 34

B.7.1 Attribute values 34
B.7.2 Trust points for certified attributes 34
B.7.3 Certification path for certified attributes 34
B.8 Rules for verification data to be followed 35
B.9 Rules for algorithm constraints and key lengths 35
B.10 Other signature policy rules 35
B.11 Signature policy protection 35
Annex C: Bibliography 36
History 39

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://webapp.etsi.org/IPR/home.asp).

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 Report (TR) 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 European Directive on a community framework for Electronic Signatures defines an electronic signature as: "data in electronic form which is attached to or logically associated with other electronic data and which serves as a method of authentication". An electronic signature as used in TS 101 733 and TS 101 903 (XAdES) (see bibliography) is a form of advanced electronic signature as defined in the Directive. An electronic signature produced in accordance with those documents provides evidence that can be processed to get confidence that some commitment has been explicitly endorsed under a Signature policy, at a given time, by a signer under an identifier, e.g. a name or a pseudonym, and optionally a role.

Although neither document does not mandate any form of Signature Policy specification, TS 101 733 originally specified an ASN.1 based syntax that may be used to define a structured Signature Policy in a way that machines can read and process. Since TS 101 733 V1.4.0 only contains the description of the formats of Electronic Signatures using the ASN.1 syntax, this document is basically an extract from the Signature Policy specification found in the original versions of TS 101 733 and using the ASN.1 syntax. At the time of publication of the present document, no implementation of the Signature Policy format using the ASN.1 syntax has been reported.

At the time of publication of the present document, two documents exist describing in detail the various components of a signature policy:

- using an XML syntax: TR 102 038 (XML format of signature policies), see bibliography;
- using an ASN.1 syntax: TR 102 272 (ASN.1 format of signature policies), i.e. the present document.

Further explanations and use of signature policies requirements for multiples signatures over a single document are published in TR 102 045 (see bibliography).

1 Scope

The present document covers the aspects of Electronic Signature Policies that were defined in TS 101 733 v1 and other older versions of that document.

No specific format is mandated for a signature policy specification. A signature policy may be specified either:

- in a free form document for human interpretation; or
- in a structured form using an agreed syntax and encoding.

The present document specifies the various components of a signature policy and one specific format using an ASN.1 syntax and DER encoding.

2 References

For the purposes of this Technical Report (TR) the following references apply:

[1] ITU-T Recommendation X.509 (1997) | ISO/IEC 9594-8 (1998): "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks".

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

[3] ITU-T Recommendation X.690 (2002) / ISO/IEC 8825-1 (2002): "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)".

[4] ITU-T Recommendation F.1 (1998): "Operational provisions for the international public telegram service".

[5] IETF RFC 3494: "Lightweight Directory Access Protocol".

[6] IETF RFC 3280: "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", see also RFC 3280 (April 2002).

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

[8] IETF RFC 3369: "Cryptographic Message Syntax".

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

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

[11] ISO/IEC 13888-1 (1997): "Information technology - Security techniques - Non-repudiation - Part 1: General".

[12] ITU-T Recommendation X.400 (1999): "Message handling services: Message handling system and service overview".

[13] ITU-T Recommendation X.500 (2001): "Information technology - Open Systems Interconnection - The Directory: Overview of concepts, models and services".

[14] ITU-T Recommendation X.501 (2001): "Information technology - Open Systems Interconnection - The Directory: Models".

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

[16] ITU-T Recommendation X.680 (2002) / ISO/IEC 8824-1 (2002): "Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation".

[17] IETF RFC 2450: "Proposed TLA and NLA Assignment Rule".

3 Definitions and abbreviations

3.1 Definitions

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

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

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

Attribute Authority Revocation List (AARL): references to attribute certificates issued to AAs, that are no longer considered valid by the issuing authority

Attribute Certificate Revocation List (ARL): 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 to an attribute authority)

cautionary period: period after the signing time that it is mandated the verifier shall wait to get high assurance of the validity of the signer's key and that any relevant revocation has been notified

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

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

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

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 [10].

public key certificate: public keys of a user, together with some other information, rendered unforgeable by encipherment with the private key of the certification authority which issued it

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

Rivest-Shamir-Adleman (RSA): highly secure cryptography method using a two-part key

signature policy: set of rules for the creation and validation of an electronic signature, under which the signature can be determined to be valid

signature policy issuer: entity that defines the technical and procedural requirements for electronic signature creation and validation, in order to meet a particular business need

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

signer: entity that creates an electronic signature

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

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 which passes validation according to a signature validation policy

verifier: entity that verifies an evidence

NOTE 1: See ISO/IEC 13888-1 [11].

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
API	Application Program Interface
ARL	Authority Revocation List
ASN.1	Abstract Syntax Notation 1
CA	Certification Authority
CAD	Card Accepting Device
CMS	Cryptographic Message Syntax
CRL	Certificate Revocation List
DER	Distinguished Encoding Rules (for ASN.1)
ES	Electronic Signature
ES-T	Electronic Signature with Timestamp
MIME	Multipurpose Internet Mail Extensions
OCSP	Online Certificate Status Provider
OID	Object Identifier
PKIX	internet X.509 Public Key Infrastructure
RSA	Rivest-Shamir-Adleman
SHA-1	Secure Hash Algorithm 1 (see annex E on cryptographic algorithms)
TSA	Time-Stamping Authority
TSP	Trusted Service Provider
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	eXtensible Markup Language

4 Signature Policy overview

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.

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

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

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

The signature policy shall include:

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

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

The formal structure for an explicit Signature Validation Policy may be expressed using various formal syntax's, including ANS.1, XML, other formats. However for a given explicit signature there shall be one definitive form that has a unique binary encoded value.

5 Signature policy specification in informal free text form

A signature policy must be identifiable and include the following identity information:

- An unambiguous identifier of the **Algorithm** used to protect the signature policy Information.
- A **hash** value of the signature policy information, which shall be re-calculated and checked whenever, the policy is passed between the issuer and signer/verifier.

In the **Signature Policy Information** section it should be specified the following information:

- The **Signature Policy Identifier** is an identifier of the signature policy that must uniquely identify the policy, and is specific to a particular version issued on the given date.
- A field that holds the **Date of Issue** for this Signature Policy.
- A field for the body responsible for issuing the Signature Policy, which is the **Signature Policy Issuer**. This may be used by the signer or verifier in deciding if a policy is to be trusted, in which case the signer/verifier shall authenticate the origin of the signature policy as coming from the identified issuer.
- A field that holds the **Field of Application** for this Signature Policy. This field holds in general terms the general legal/contract/application contexts in which the signature policy is to be used and the specific purposes for which the electronic signature is to be applied.
- It can be optionally included a **Signature Policy Extensions** section, where it can be declared any other information related to this Signature Policy.

In the **Signature Policy Information** section, it should also be included the **Signature Validation Policy** which defines for the signer, the data elements that shall be present in the electronic signature that is provided, and for the verifier, the data elements that shall be present under that signature policy for an electronic signature to be potentially valid. In more details, in the **Signature Validation Policy** information section, it should be specified the following information:

- A field that holds the **Signing Period** over which the signature policy may be used to generate electronic signatures, which defines the start time and date, and optionally the end time and date.
- A section that defines the **Common Rules**, which defines rules and conditions, that is common to all commitment types.
- A section that defines the **Commitment Rules**, which consists of the validation rules and conditions, which apply to given commitment types.
- It can be optionally included a **Signature Validation Policy Extensions** section, where it can be defined any other information related to this Signature Validation Policy.

Both **Common Rules** and **Commitment Rules** are defined in terms of rules for the signer or the verifier, and in terms of trust conditions for certificates, timestamps and attributes, along with any constraints on attributes that may be included in the electronic signature, and their sections contain the same set of information:

- A set of **Rules** for:
  - The Signer.
  - The Verifier.

- A set of Trust Conditions for:
  - Signing Certificate.
  - Time-stamping.
  - Attributes.
- A set of **Algorithm Constraints** can be optionally included (if constraints are required).
- It can be optionally included an **Extensions** section where it can be defined any other information related to either Common Rules or Commitment Rules.

Both **Common Rules** and **Commitment Rules** sections may contain the same set of information, but the following rules and conditions apply:

- In the Common Rules the **Common Extensions** section holds information related to the Common Rules.
- In the Commitment Rules the **Commitment Extensions** section holds information related to the Commitment Rules.
- In the Commitment Rules it should be also specified a unique **Commitment Type Identifier**, which defines the Commitment Type.
- In the Commitment Rules it should be optionally specified the **Application Field** and the **Semantics Field**, which define the specific use and meaning of the commitment within the overall field of application, defined for the policy.
- If rules and conditions are present in Common Rules**,** then the equivalent rules and conditions shall not be present in any of the Commitment Rules.
- If the Signer Rules, the Verifier Rules, the Signing Certificates Trust Conditions, and the Timestamping Trust Conditions are not present in Common Rules, then they shall be present in each Commitment Rule.

In more details, in the **Signer Rules** and the **Verifier Rules** sections, rules should be specified to identify various information the signer or the verifier require:

- The rules should identify if the **Signed Data Hash** that is used to calculate the signature, is internal or external to CMS structure.
- A set of **Signed Attributes** that shall be present under this policy and shall be provided by the signer. It should include object identifiers for all signed attributes required by this policy.
- A set of **Unsigned Attributes** that shall be present under this policy and shall be provided by the signer. If not added by the signer, they will be added by the verifier. It should include object identifiers for all unsigned attributes required by this policy. For example, if the signer requires a signature timestamp the object identifier for this attribute shall be included.
- The **Signing Certificate** attribute that shall be provided by the signer under this policy. This should identify whether the signer shall provide just the signer's certificate, or the entire full certificate path.
- It can be optionally included an **Extensions** section where it can be defined any other information related to the Signer or the Verifier Rules.

The **Signing Certificate Trust Condition**, **Time Stamp Trust Condition** and **Attribute Trust Condition** make use of the following **Certificate Requirements**, which is used to define policy for validating the signing certificate, the TSA's certificate and attribute certificates.

The **Certificate Requirements** specify information that identifies the trust points used to start (or end) certificate path processing, e.g. using a set of self signed certificates, and the initial conditions for certificate path validation. In the **Certificate Requirements** section it should be specified the following information:

- A field for the **Trust Point**. This is the specification of the trust point for the start of processing of the certificate path that gives the self-signed certificate for the CA.

- A field for the **Certificate Path Length**. This is information about the maximum number of CA certificates that may be in a certification path following the trust point. The length of the path can specify this information.

- A field for the **Acceptable Certificate Policies**. This is information about certificate policies, any of which are acceptable under the signature policy.

- A field for the **Naming Constraints**. This is the indication of a name space within which all subject names in subsequent certificates in a certification path shall be located. Restrictions may apply to the subject-distinguished name or subject alternative names. Restrictions apply only when the specified name form is present. If no name of the type is in the certificate, the certificate is acceptable. Restrictions are defined in terms of permitted or excluded name subtrees. Any name matching a restriction in the excluded subtrees is invalid regardless of information appearing in the permitted subtrees.

- A field for the **Explicit Indication** of the certificate policy. This is specification of requirement for explicit indication of the certificate policy and/or the constraints on policy mapping.

The **Signing Certificate Trust Condition**, **Time Stamp Trust Condition** and **Attribute Trust Condition** make use of the following **Revocation Requirements**, which are used to define policy for checking the revocation status of the signing certificate, the TSA's certificate and attribute certificates. In the **Revocation Requirements** section it should be specified the following information:

- A set of information for the **End Certificate Revocation Requirements**. This is specification for checks required on the leaf certificate (i.e. the signers certificate, the attribute certificate or the time-stamping authority certificate).

- A set of information for the **CA Certificate Revocation Requirements**. This is specification for checks required on CA certificates from the certification path.

- For each of the above sets of information the following fields should be included:

  - A field for the **CRL Check**. This is information for checks required whether full CRLs (or full authority revocation lists) have to be collected.

  - A field for the **OCSP Check**. This is information for OCSP responses that have to be collected.

  - A field for the **Delta CRL Check**. This is information for checks required whether delta-CRLs and the relevant associated full CRLs (or full Authority Revocation Lists) are to be collected.

  - A field for the **Other Check**. This is extension information for checks required whether any other available revocation information has to be collected.

The **Signing Certificate Trust Condition** identifies trust conditions for processing the certificate path used to validate the signing certificate. It should include the following information:

- Certificate Requirements; and

- Revocation Requirements.

The **Time Stamp Trust Condition** identifies trust conditions for processing the certificate path used to authenticate the time-stamping authority and constraints on the name of the time-stamping authority. It should include the following information:

- A field for the **Time-stamping Authorities Public Key Rules** of the time-stamping authorities. This information specifies if any rules apply to the certification of the time-stamping authorities public key.

- A field for the **Time-stamp Revocation Requirements** that is used to check the revocation status of the time stamp. This information defines minimum requirements for revocation information and is obtained through CRLs and/or OCSP responses. This information could include the type of checks that should be carried out and whether these checks are needed or not. These checks shall be carried out once the cautionary period is over.

- A field for any additional **Naming Constraints** on the trusted time-stamping authority.

- A field for a defined **Cautionary period**. This is the period after the signing time that it is mandated the verifier shall wait to get high assurance of the validity of the signer's key and that any relevant revocation has been notified.

- A field for the **Maximum Acceptable Time**. This is the time between the signing time and the time at which the signature timestamp is created for the verifier.

The **Attribute Trust Condition** must be present so any certified attributes can be considered to be valid under this validation policy. It should include the following information:

- A field for the **Signer Attributes**. This is information about the "claimed" or "certified" attributes of the signer.

- A field for the **Attribute Certificate Conditions**. This information specifies the certificate path conditions for any attribute certificate.

- A field for the **Attribute Revocation Requirements** that is used to check the revocation status of Attribute Certificates, if any are present. This information defines minimum requirements for revocation information and is obtained through CRLs and/or OCSP responses. This information could include the type of checks that should be carried out and whether these checks are needed or not.

- A field for the **Attribute Constraints** can be optionally included (if constraints are required). This is information about constraints on the specific attribute types and their values that may be validated under this policy.

A set of **Algorithm Constraints** can be optionally included (if constraints are required). There are different types of constraints:

- Signer Algorithm Constraints.

- Issuer of End Entity Certificates Algorithm Constraints.

- Issuer of CA Certificates Algorithm Constraints.

- Attribute Authority Algorithm Constraints.

- Time-stamping Authority Algorithm Constraints.

This set of **Algorithm Constraints** is optionally included and for each type of these constraints it should be identified the following information:

- The **Signing Algorithms** (hash, public key cryptography, combined hash and public key cryptography) that may be used for specific purposes.

- A field for the **Minimum Key Length** that is required for these Signing Algorithms.

- It can be optionally included an **Extensions** section where it can be defined any other information related to the Algorithm Constraints.

6 Signature policy specification in ASN.1

A signature policy specification in ASN.1 shall:

- be identifiable by an Object Identifier;

- have a unique encoding form so that a single hash value can be computed over it. For this reason the use of DER is mandated.

A signature policy specification includes general information about the policy, the validation policy rules and other signature policy information. Annex B provide more information on what can be included in a signature policy.

6.1 Overall ASN.1 structure

The overall structure of a signature policy defined using ASN.1 is given in this clause. This ASN.1 syntax is encoded using the distinguished encoding rules.

In this structure the policy information is preceded by an identifier for the hashing algorithm used to protect the signature policy and followed by the hash value which shall be re-calculated and checked whenever the policy is passed between the issuer and signer/verifier. The hash is calculated without the outer type and length fields.

```
SignaturePolicy ::= SEQUENCE {
    signPolicyHashAlg      AlgorithmIdentifier,
    signPolicyInfo         SignPolicyInfo,
    signPolicyHash         SignPolicyHash     OPTIONAL }

SignPolicyHash ::= OCTET STRING

SignPolicyInfo ::= SEQUENCE {
    signPolicyIdentifier            SignPolicyId,
    dateOfIssue                     GeneralizedTime,
    policyIssuerName                PolicyIssuerName,
    fieldOfApplication              FieldOfApplication,
    signatureValidationPolicy       SignatureValidationPolicy,
    signPolExtensions               SignPolExtensions       OPTIONAL
    }

SignPolicyId ::= OBJECT IDENTIFIER
```

The **policyIssuerName** field identifies the policy issuer in one or more of the general name forms.

```
PolicyIssuerName ::= GeneralNames
```

The **fieldofApplication** is a description of the expected application of this policy.

```
FieldOfApplication ::= DirectoryString
```

The signature validation policy rules are fully processable to allow the validation of electronic signatures issued under that signature policy. They are described in the rest of this clause.

6.2 Signature validation policy

The signature validation policy defines for the signer which data elements shall be present in the electronic signature he provides and for the verifier which data elements shall be present under that signature policy for an electronic signature to be potentially valid.

The signature validation policy is described as follows:

```
SignatureValidationPolicy ::= SEQUENCE {
    signingPeriod          SigningPeriod,
    commonRules            CommonRules,
    commitmentRules        CommitmentRules,
    signPolExtensions      SignPolExtensions        OPTIONAL
     }
```

The **signingPeriod** identifies the date and time before which the signature policy should not be used for creating signatures, and an optional date after which it should not be used for creating signatures.

```
SigningPeriod ::= SEQUENCE {
    notBefore   GeneralizedTime,
    notAfter    GeneralizedTime OPTIONAL }
```

6.3 Common Rules

The **CommonRules** define rules that are common to all commitment types. These rules are defined in terms of trust conditions for certificates, timestamps and attributes, along with any constraints on attributes that may be included in the electronic signature.

```
CommonRules  ::= SEQUENCE {
    signerAndVeriferRules          [0]  SignerAndVerifierRules     OPTIONAL,
    signingCertTrustCondition      [1]  SigningCertTrustCondition   OPTIONAL,
    timeStampTrustCondition        [2]  TimestampTrustCondition     OPTIONAL,
    attributeTrustCondition        [3]  AttributeTrustCondition     OPTIONAL,
    algorithmConstraintSet         [4]  AlgorithmConstraintSet      OPTIONAL,
    signPolExtensions              [5]  SignPolExtensions           OPTIONAL
    }
```

If a field is present in CommonRules then the equivalent field shall not be present in any of the CommitmentRules (see below). If any of the following fields are not present in CommonRules then it shall be present in each CommitmentRule:

- signerAndVeriferRules;
- signingCertTrustCondition;
- timeStampTrustCondition.

6.4 Commitment Rules

The **CommitmentRules** consists of the validation rules which apply to given commitment types:

```
CommitmentRules ::= SEQUENCE OF CommitmentRule
```

The **CommitmentRule** for given commitment types are defined in terms of trust conditions for certificates, timestamps and attributes, along with any constraints on attributes that may be included in the electronic signature.

```
CommitmentRule  ::= SEQUENCE {
    selCommitmentTypes                 SelectedCommitmentTypes,
    signerAndVeriferRules          [0]  SignerAndVerifierRules      OPTIONAL,
    signingCertTrustCondition      [1]  SigningCertTrustCondition   OPTIONAL,
    timeStampTrustCondition        [2]  TimestampTrustCondition     OPTIONAL,
    attributeTrustCondition        [3]  AttributeTrustCondition     OPTIONAL,
    algorithmConstraintSet         [4]  AlgorithmConstraintSet      OPTIONAL,
    signPolExtensions              [5]  SignPolExtensions           OPTIONAL
    }

SelectedCommitmentTypes ::= SEQUENCE OF CHOICE {
    empty                        NULL,
    recognizedCommitmentType     CommitmentType }
```

If the SelectedCommitmentTypes indicates "*empty*" then this rule applied when a commitment type is not present (i.e. the type of commitment is indicated in the semantics of the message). Otherwise, the electronic signature shall contain a commitment type indication that shall fit one of the commitments types that are mentioned in CommitmentType.

A specific commitment type identifier shall not appear in more than one commitment rule.

```
CommitmentType ::= SEQUENCE {
    identifier          CommitmentTypeIdentifier,
    fieldOfApplication  [0] FieldOfApplication OPTIONAL,
    semantics           [1] DirectoryString OPTIONAL }
```

The **fieldOfApplication** and **semantics** fields define the specific use and meaning of the commitment within the overall field of application defined for the policy.

6.5 Signer and Verifier Rules

The SignerAndVerifierRules consists of signer rule and verification rules as defined below:

```
SignerAndVerifierRules ::= SEQUENCE {
    signerRules      SignerRules,
    verifierRules    VerifierRules }
```

6.5.1 Signer rules

The signer rules identify:

- if the **eContent** is empty and the signature is calculated using a hash of signed data external to CMS structure;
- the CMS signed attributes that shall be provided by the signer under this policy;
- the CMS unsigned attribute that shall be provided by the signer under this policy;
- whether the certificate identifiers from the full certification path up to the trust point shall be provided by the signer in the **SigningCertificate** attribute;
- whether a signer's certificate, or all certificates in the certification path to the trust point shall be provided by the signer in the **certificates** field of **SignedData.**

```
SignerRules ::= SEQUENCE {
    externalSignedData         BOOLEAN  OPTIONAL,
            -- True if signed data is external to CMS structure
            -- False if signed data part of CMS structure
            -- not present if either allowed
    mandatedSignedAttr         CMSAttrs,    -- Mandated CMS signed attributes
    mandatedUnsignedAttr       CMSAttrs,    -- Mandated CMS unsigned attributed
    mandatedCertificateRef     [0] CertRefReq DEFAULT signerOnly,
                -- Mandated Certificate Reference
    mandatedCertificateInfo    [1] CertInfoReq DEFAULT none,
                -- Mandated Certificate Info
    signPolExtensions        [2] SignPolExtensions      OPTIONAL
                          }

CMSAttrs ::= SEQUENCE OF OBJECT IDENTIFIER
```

The **mandatedSignedAttr** field shall include the object identifier for all those signed attributes required by the present document as well as additional attributes required by this policy.

The **mandatedUnsignedAttr** field shall include the object identifier for all those unsigned attributes required by the present document as well as additional attributes required this policy. For example, if a signature timestamp (see clause 1.1) is required *by the signer* the object identifier for this attribute shall be included.

The **mandatedCertificateRef** identifies whether just the signer's certificate, or all the full certificate path shall be provided by the signer.

```
CertRefReq ::= ENUMERATED {
                signerOnly (1),     -- Only reference to signer cert mandated
                fullPath (2)
                        -- References for full cert path up to a trust point required
                                   }
```

The **mandatedCertificateInfo** field identifies whether a signer's certificate, or all certificates in the certification path to the trust point shall be provided by the signer in the **certificates** field of **SignedData.**

```
CertInfoReq ::= ENUMERATED {
                none (0)    ,           -- No mandatory requirements
                signerOnly (1)  ,       -- Only reference to signer cert mandated
                fullPath (2)
                        -- References for full cert path up to a trust point mandated
        }
```

6.5.2 Verifier rules

The verifier rules identify:

- The CMS unsigned attributes that shall be present under this policy and shall be added by the verifier if not added by the signer.

```
VerifierRules ::= SEQUENCE {
        mandatedUnsignedAttr    MandatedUnsignedAttr,
        signPolExtensions       SignPolExtensions       OPTIONAL
        }

MandatedUnsignedAttr ::=  CMSAttrs    -- Mandated CMS unsigned attributed
```

6.6 Certificate and revocation requirement

The **SigningCertTrustCondition**, **TimestampTrustCondition** and **AttributeTrustCondition** (defined in subsequent clauses) make use of two ASN1 structures which are defined below: **CertificateTrustTrees** and **CertRevReq**.

6.6.1 Certificate requirements

The **certificateTrustTrees** identifies a set of self signed certificates for the trust points used to start (or end) certificate path processing and the initial conditions for certificate path validation as defined RFC 2459 [6] clause 6. This ASN1 structure is used to define policy for validating the signing certificate, the TSA's certificate and attribute certificates.

```
CertificateTrustTrees ::=   SEQUENCE OF CertificateTrustPoint

CertificateTrustPoint ::= SEQUENCE {
    trustpoint              Certificate,                    -- self-signed certificate
    pathLenConstraint   [0] PathLenConstraint   OPTIONAL,
    acceptablePolicySet [1] AcceptablePolicySet OPTIONAL,  -- If not present "any policy"
    nameConstraints     [2] NameConstraints     OPTIONAL,
    policyConstraints   [3] PolicyConstraints   OPTIONAL }
```

The **trustPoint** field gives the self signed certificate for the CA that is used as the trust point for the start of certificate path processing.

The **pathLenConstraint** field gives the maximum number of CA certificates that may be in a certification path following the **trustpoint**. A value of zero indicates that only the given **trustpoint** certificate and an end-entity certificate may be used. If present, the pathLenConstraint field shall be greater than or equal to zero. Where pathLenConstraint is not present, there is no limit to the allowed length of the certification path.

```
PathLenConstraint    ::=   INTEGER (0..MAX)
```

The **acceptablePolicySet** field identifies the initial set of certificate policies, any of which are acceptable under the signature policy.

```
AcceptablePolicySet ::= SEQUENCE OF CertPolicyId

CertPolicyId ::= OBJECT IDENTIFIER
```

The **nameConstraints** field indicates a name space within which all subject names in subsequent certificates in a certification path shall be located. Restrictions may apply to the subject distinguished name or subject alternative names. Restrictions apply only when the specified name form is present. If no name of the type is in the certificate, the certificate is acceptable.

Restrictions are defined in terms of permitted or excluded name subtrees. Any name matching a restriction in the **excludedSubtrees** field is invalid regardless of information appearing in the **permittedSubtrees**.

```
 NameConstraints ::= SEQUENCE {
          permittedSubtrees       [0]     GeneralSubtrees OPTIONAL,
          excludedSubtrees        [1]     GeneralSubtrees OPTIONAL }

      GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree

      GeneralSubtree ::= SEQUENCE {
           base                    GeneralName,
           minimum         [0]     BaseDistance DEFAULT 0,
           maximum         [1]     BaseDistance OPTIONAL }

      BaseDistance ::= INTEGER (0..MAX)
```

The **policyConstraints** extension constrains path processing in two ways. It can be used to prohibit policy mapping or require that each certificate in a path contain an acceptable policy identifier.

The **policyConstraints** field, if present specifies requirement for explicit indication of the certificate policy and/or the constraints on policy mapping.

```
PolicyConstraints ::= SEQUENCE {
        requireExplicitPolicy           [0] SkipCerts OPTIONAL,
        inhibitPolicyMapping            [1] SkipCerts OPTIONAL }

SkipCerts ::= INTEGER (0..MAX)
```

If the **inhibitPolicyMapping** field is present, the value indicates the number of additional certificates that may appear in the path (including the trustpoint's self certificate) before policy mapping is no longer permitted. For example, a value of one indicates that policy mapping may be processed in certificates issued by the subject of this certificate, but not in additional certificates in the path.

If the **requireExplicitPolicy** field is present, subsequent certificates shall include an acceptable policy identifier. The value of **requireExplicitPolicy** indicates the number of additional certificates that may appear in the path (including the trustpoint's self certificate) before an explicit policy is required. An acceptable policy identifier is the identifier of a policy required by the user of the certification path or the identifier of a policy which has been declared equivalent through policy mapping.

6.6.2 Revocation requirements

The **RevocRequirements** field specifies minimum requirements for revocation information, obtained through CRLs and/or OCSP responses, to be used in checking the revocation status of certificates. This ASN1 structure is used to define policy for validating the signing certificate, the TSA's certificate and attribute certificates.

```
CertRevReq ::= SEQUENCE {
    endCertRevReq   RevReq,
    caCerts         [0] RevReq
      }
```

Certificate revocation requirements are specified in terms of checks required on:

- **endCertRevReq**: end certificates (i.e. the signers certificate, the attribute certificate or the timestamping authority certificate);
- **caCerts**: CA certificates.

```
RevReq ::= SEQUENCE  {
    enuRevReq  EnuRevReq,
    exRevReq    SignPolExtensions OPTIONAL}

EnuRevReq  ::= ENUMERATED {
    clrCheck    (0), --Checks shall be made against current CRLs
            -- (or authority revocation lists)
    ocspCheck   (1), -- The revocation status shall be checked
            -- using the Online Certificate Status Protocol (RFC 2450)
    bothCheck   (2),    -- Both CRL and OCSP checks shall be carried out
    eitherCheck (3),    -- At least one of CRL or OCSP checks shall be carried out
    noCheck     (4),    -- no check is mandated
```

```
    other       (5)     -- Other mechanism as defined by signature policy extension
      }
```

Revocation requirements are specified in terms of:

- **clrCheck:** Checks shall be made against current CRLs (or authority revocation lists);
- **ocspCheck:** The revocation status shall be checked using the Online Certificate Status Protocol (RFC 2450 [17]);
- **bothCheck:** Both OCSP and CRL checks shall be carried out;
- **eitherCheck:** Either OCSP or CRL checks shall be carried out;
- **noCheck:** No check is mandated.

6.7 Signing certificate trust conditions

The **SigningCertTrustCondition** field identifies trust conditions for certificate path processing used to validate the signing certificate.

```
SigningCertTrustCondition ::=   SEQUENCE {
     signerTrustTrees              CertificateTrustTrees,
     signerRevReq                  CertRevReq
}
```

6.8 Time-Stamp trust conditions

The **TimeStampTrustCondition** field identifies trust conditions for certificate path processing used to authenticate the timstamping authority and constraints on the name of the timestamping authority. This applies to the timestamp that shall be present in every ES-T.

```
TimestampTrustCondition ::= SEQUENCE {
    ttsCertificateTrustTrees    [0]     CertificateTrustTrees    OPTIONAL,
    ttsRevReq                   [1]     CertRevReq              OPTIONAL,
    ttsNameConstraints          [2]     NameConstraints         OPTIONAL,
    cautionPeriod               [3]     DeltaTime               OPTIONAL,
    signatureTimestampDelay     [4]     DeltaTime               OPTIONAL }

DeltaTime ::= SEQUENCE {
    deltaSeconds    INTEGER,
    deltaMinutes    INTEGER,
    deltaHours      INTEGER,
    deltaDays       INTEGER }
```

If **ttsCertificateTrustTrees** is not present then the same rule as defined in **certificateTrustCondition** applies to certification of the timestamping authorities public key.

The **tstrRevReq** specifies minimum requirements for revocation information, obtained through CRLs and/or OCSP responses, to be used in checking the revocation status of the time stamp that shall be present in the ES-T.

If **ttsNameConstraints** is not present then there are no additional naming constraints on the trusted timestamping authority other than those implied by the **ttsCertificateTrustTrees**.

The **cautionPeriod** field specifies a caution period after the signing time that it is mandated the verifier shall wait to get high assurance of the validity of the signer's key and that any relevant revocation has been notified. The revocation status information forming the ES with Complete validation data shall not be collected and used to validate the electronic signature until after this caution period.

The **signatureTimestampDelay** field specifies a maximum acceptable time between the signing time and the time at which the signature timestamp, as used to form the ES Timestamped, is created for the verifier. If the signature timestamp is later that the time in the signing-time attribute by more than the value given in **signatureTimestampDelay**, the signature shall be considered invalid.

6.9 Attribute trust conditions

If the **attributeTrustCondition** field is not present then any certified attributes may not considered to be valid under this validation policy.

The **AttributeTrustCondition** field is defined as follows:

```
AttributeTrustCondition ::= SEQUENCE {
    attributeMandated            BOOLEAN,              -- Attribute shall be present
    howCertAttribute             HowCertAttribute,
    attrCertificateTrustTrees   [0] CertificateTrustTrees   OPTIONAL,
    attrRevReq                  [1] CertRevReq              OPTIONAL,
    attributeConstraints        [2] AttributeConstraints   OPTIONAL }
```

If **attributeMandated** is true then an attribute, certified within the following constraints, shall be present. If false, then the signature is still valid if no attribute is specified.

The **howCertAttribute** field specifies whether attributes uncertified attributes "claimed" by the signer, or certified in an attribute certificate or either using the signer attributes attribute defined in TS 101 733.

```
HowCertAttribute ::= ENUMERATED {
    claimedAttribute    (0),
    certifiedAttribtes  (1),
    either              (2) }
```

The **attrCertificateTrustTrees** specifies certificate path conditions for any attribute certificate. If not present the same rules apply as in **certificateTrustCondition**.

The **attrRevReq** specifies minimum requirements for revocation information, obtained through CRLs and/or OCSP responses, to be used in checking the revocation status of Attribute Certificates, if any are present.

If the **attributeConstraints** field is not present then there are no constraints on the attributes that may be validated under this policy. The **attributeConstraints** field is defined as follows:

```
AttributeConstraints ::= SEQUENCE {
    attributeTypeConstraints    [0] AttributeTypeConstraints  OPTIONAL,
    attributeValueConstraints   [1] AttributeValueConstraints OPTIONAL }
```

If present, the attributeTypeConstraints field specifies the attribute types which are considered valid under the signature policy. Any value for that attribute is considered valid.

```
AttributeTypeConstraints ::= SEQUENCE OF AttributeType
```

If present, the attributeTypeConstraints field specifies the specific attribute values which are considered valid under the signature policy.

```
AttributeValueConstraints ::= SEQUENCE OF AttributeTypeAndValue
```

6.10 Algorithm constraints

The **algorithmConstrains** fields, if present, identifies the signing algorithms (hash, public key cryptography, combined hash and public key cryptography) that may be used for specific purposes and any minimum length. If this field is not present then the policy applies no constraints.

```
AlgorithmConstraintSet ::= SEQUENCE {   -- Algorithm constrains on:
signerAlgorithmConstraints  [0]     AlgorithmConstraints OPTIONAL, -- signer
eeCertAlgorithmConstraints  [1]     AlgorithmConstraints OPTIONAL, -- issuer of end entity certs.
caCertAlgorithmConstraints  [2]     AlgorithmConstraints OPTIONAL, -- issuer of CA certificates
aaCertAlgorithmConstraints  [3]     AlgorithmConstraints OPTIONAL, -- Attribute Authority
tsaCertAlgorithmConstraints [4]     AlgorithmConstraints OPTIONAL  -- TimeStamping Authority
            }
```

```
AlgorithmConstraints ::= SEQUENCE OF AlgAndLength

AlgAndLength ::= SEQUENCE {
    algID           OBJECT IDENTIFIER,
    minKeyLength    INTEGER     OPTIONAL, -- Minimum key length in bits
    other       SignPolExtensions OPTIONAL
         }
```

6.11 Signature policy extensions

Additional signature policy rules may be added to:

- the overall signature policy structure, as defined in clause 6.1;
- the signature validation policy structure, as defined in clause 6.2;
- the common rules, as defined in clause 6.3;
- the commitment rules, as defined in clause 6.4;
- the signer rules, as defined in clause 6.5.1;
- the verifier rules, as defined in clause 6.5.2;
- the revocation requirements in clause 6.6.2;
- the algorithm constraints in clause 6.10.

These extensions to the signature policy rules shall be defined using an ASN.1 syntax with an associated object identifier carried in the **SignPolExtn** as defined below:

```
SignPolExtensions ::= SEQUENCE OF SignPolExtn

SignPolExtn ::= SEQUENCE {
        extnID      OBJECT IDENTIFIER,
        extnValue   OCTET STRING  }
```

The **extnID** field shall contain the object identifier for the extension. The **extnValue** field shall contain the DER (see ITU-T Recommendation X.690 [3]) encoded value of the extension. The definition of an extension, as identified by **extnID** shall include a definition of the syntax and semantics of the extension.

Annex A:
ASN.1 modules

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

A.1 Signature policies definitions using X.208 (1988) ASN.1 syntax

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

```
ETS-ElectronicSignaturePolicies-88syntax { iso(1) member-body(2)
    us(840) rsadsi(113549) pkcs(1) pkcs-9(9) smime(16) id-mod(0) 7}


DEFINITIONS EXPLICIT TAGS ::=
BEGIN
-- EXPORTS All

IMPORTS


-- Internet X.509 Public Key Infrastructure - Certificate and CRL Profile: RFC 2459
    Certificate, AlgorithmIdentifier, CertificateList, Name, GeneralNames, GeneralName,
    DirectoryString,Attribute, AttributeTypeAndValue, AttributeType, AttributeValue,
     PolicyInformation, BMPString, UTF8String

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

-- The structures may also be imported from :
-- PKIX1Explicit88 { iso(1) identified-organization(3) dod(6) internet(1)
-- security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18) }


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

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

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


-- Signature Policy Specification
-- ==============================

SignaturePolicy ::= SEQUENCE {
    signPolicyHashAlg      AlgorithmIdentifier,
    signPolicyInfo         SignPolicyInfo,
    signPolicyHash         SignPolicyHash     OPTIONAL }

SignPolicyHash ::= OCTET STRING
```

```
SignPolicyInfo ::= SEQUENCE {
    signPolicyIdentifier            SignPolicyId,
    dateOfIssue                     GeneralizedTime,
    policyIssuerName                PolicyIssuerName,
    fieldOfApplication              FieldOfApplication,
    signatureValidationPolicy       SignatureValidationPolicy,
    signPolExtensions               SignPolExtensions       OPTIONAL
    }

SignPolicyId ::= OBJECT IDENTIFIER

PolicyIssuerName ::= GeneralNames

FieldOfApplication ::= DirectoryString

SignatureValidationPolicy ::= SEQUENCE {
    signingPeriod          SigningPeriod,
    commonRules            CommonRules,
    commitmentRules        CommitmentRules,
    signPolExtensions      SignPolExtensions        OPTIONAL
     }

SigningPeriod ::= SEQUENCE {
    notBefore   GeneralizedTime,
    notAfter    GeneralizedTime OPTIONAL }

CommonRules  ::= SEQUENCE {
    signerAndVeriferRules          [0]  SignerAndVerifierRules      OPTIONAL,
    signingCertTrustCondition      [1]  SigningCertTrustCondition   OPTIONAL,
    timeStampTrustCondition        [2]  TimestampTrustCondition     OPTIONAL,
    attributeTrustCondition        [3]  AttributeTrustCondition     OPTIONAL,
    algorithmConstraintSet         [4]  AlgorithmConstraintSet      OPTIONAL,
    signPolExtensions              [5]  SignPolExtensions           OPTIONAL
    }

CommitmentRules ::= SEQUENCE OF CommitmentRule

CommitmentRule  ::= SEQUENCE {
    selCommitmentTypes                  SelectedCommitmentTypes,
    signerAndVeriferRules          [0]  SignerAndVerifierRules      OPTIONAL,
    signingCertTrustCondition      [1]  SigningCertTrustCondition   OPTIONAL,
    timeStampTrustCondition        [2]  TimestampTrustCondition     OPTIONAL,
    attributeTrustCondition        [3]  AttributeTrustCondition     OPTIONAL,
    algorithmConstraintSet         [4]  AlgorithmConstraintSet      OPTIONAL,
    signPolExtensions              [5]  SignPolExtensions           OPTIONAL
    }

SelectedCommitmentTypes ::= SEQUENCE OF CHOICE {
    empty                       NULL,
    recognizedCommitmentType    CommitmentType }

CommitmentType ::= SEQUENCE {
    identifier          CommitmentTypeIdentifier,
    fieldOfApplication  [0] FieldOfApplication OPTIONAL,
    semantics           [1] DirectoryString OPTIONAL }

SignerAndVerifierRules ::= SEQUENCE {
    signerRules      SignerRules,
    verifierRules    VerifierRules }

SignerRules ::= SEQUENCE {
    externalSignedData         BOOLEAN  OPTIONAL,
            -- True if signed data is external to CMS structure
            -- False if signed data part of CMS structure
            -- not present if either allowed
    mandatedSignedAttr         CMSAttrs,    -- Mandated CMS signed attributes
    mandatedUnsignedAttr       CMSAttrs,    -- Mandated CMS unsigned attributed
    mandatedCertificateRef     [0] CertRefReq DEFAULT signerOnly,
                -- Mandated Certificate Reference
    mandatedCertificateInfo    [1] CertInfoReq DEFAULT none,
                -- Mandated Certificate Info
    signPolExtensions        [2] SignPolExtensions      OPTIONAL
                        }

CMSAttrs ::= SEQUENCE OF OBJECT IDENTIFIER

CertRefReq ::= ENUMERATED {
                signerOnly (1),     -- Only reference to signer cert mandated
```

```
                fullPath (2)
                        -- References for full cert path up to a trust point required
                                }

CertInfoReq ::= ENUMERATED {
                none (0)    ,           -- No mandatory requirements
                signerOnly (1)  ,       -- Only reference to signer cert mandated
                fullPath (2)
                        -- References for full cert path up to a trust point mandated
        }

VerifierRules ::= SEQUENCE {
        mandatedUnsignedAttr    MandatedUnsignedAttr,
        signPolExtensions       SignPolExtensions       OPTIONAL
        }

MandatedUnsignedAttr ::=  CMSAttrs    -- Mandated CMS unsigned attributed


CertificateTrustTrees ::=   SEQUENCE OF CertificateTrustPoint

CertificateTrustPoint ::= SEQUENCE {
    trustpoint              Certificate,                        -- self-signed certificate
    pathLenConstraint   [0] PathLenConstraint   OPTIONAL,
    acceptablePolicySet [1] AcceptablePolicySet OPTIONAL,  -- If not present "any policy"
    nameConstraints     [2] NameConstraints     OPTIONAL,
    policyConstraints   [3] PolicyConstraints   OPTIONAL }

PathLenConstraint    ::=   INTEGER (0..MAX)

AcceptablePolicySet ::= SEQUENCE OF CertPolicyId

CertPolicyId ::= OBJECT IDENTIFIER

NameConstraints ::= SEQUENCE {
           permittedSubtrees       [0]     GeneralSubtrees OPTIONAL,
           excludedSubtrees        [1]     GeneralSubtrees OPTIONAL }

GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree

GeneralSubtree ::= SEQUENCE {
           base                    GeneralName,
           minimum         [0]     BaseDistance DEFAULT 0,
           maximum         [1]     BaseDistance OPTIONAL }

BaseDistance ::= INTEGER (0..MAX)

PolicyConstraints ::= SEQUENCE {
        requireExplicitPolicy           [0] SkipCerts OPTIONAL,
        inhibitPolicyMapping            [1] SkipCerts OPTIONAL }

SkipCerts ::= INTEGER (0..MAX)

CertRevReq ::= SEQUENCE {
    endCertRevReq   RevReq,
    caCerts         [0] RevReq
      }

RevReq ::= SEQUENCE  {
    enuRevReq  EnuRevReq,
    exRevReq    SignPolExtensions OPTIONAL}

EnuRevReq  ::= ENUMERATED {
    clrCheck    (0), --Checks shall be made against current CRLs
            -- (or authority revocation lists)
    ocspCheck   (1), -- The revocation status shall be checked
            -- using the Online Certificate Status Protocol (RFC 2450)
    bothCheck   (2),    -- Both CRL and OCSP checks shall be carried out
    eitherCheck (3),    -- At least one of CRL or OCSP checks shall be carried out
    noCheck     (4),    -- no check is mandated
    other       (5)     -- Other mechanism as defined by signature policy extension
      }

SigningCertTrustCondition ::=   SEQUENCE {
     signerTrustTrees              CertificateTrustTrees,
     signerRevReq                  CertRevReq
}
```

```
TimestampTrustCondition ::= SEQUENCE {
    ttsCertificateTrustTrees    [0]     CertificateTrustTrees    OPTIONAL,
    ttsRevReq                   [1]     CertRevReq              OPTIONAL,
    ttsNameConstraints          [2]     NameConstraints         OPTIONAL,
    cautionPeriod               [3]     DeltaTime               OPTIONAL,
    signatureTimestampDelay     [4]     DeltaTime               OPTIONAL }

DeltaTime ::= SEQUENCE {
    deltaSeconds    INTEGER,
    deltaMinutes    INTEGER,
    deltaHours      INTEGER,
    deltaDays       INTEGER }

AttributeTrustCondition ::= SEQUENCE {
    attributeMandated            BOOLEAN,              -- Attribute shall be present
    howCertAttribute             HowCertAttribute,
    attrCertificateTrustTrees   [0] CertificateTrustTrees   OPTIONAL,
    attrRevReq                  [1] CertRevReq             OPTIONAL,
    attributeConstraints        [2] AttributeConstraints   OPTIONAL }


HowCertAttribute ::= ENUMERATED {
    claimedAttribute    (0),
    certifiedAttribtes  (1),
    either              (2) }

AttributeConstraints ::= SEQUENCE {
    attributeTypeConstraints    [0] AttributeTypeConstraints  OPTIONAL,
    attributeValueConstraints   [1] AttributeValueConstraints OPTIONAL }


AttributeTypeConstraints ::= SEQUENCE OF AttributeType

AttributeValueConstraints ::= SEQUENCE OF AttributeTypeAndValue


AlgorithmConstraintSet ::= SEQUENCE {   -- Algorithm constrains on:
signerAlgorithmConstraints  [0]     AlgorithmConstraints OPTIONAL, -- signer
eeCertAlgorithmConstraints  [1]     AlgorithmConstraints OPTIONAL, -- issuer of end entity certs.
caCertAlgorithmConstraints  [2]     AlgorithmConstraints OPTIONAL, -- issuer of CA certificates
aaCertAlgorithmConstraints  [3]     AlgorithmConstraints OPTIONAL, -- Attribute Authority
tsaCertAlgorithmConstraints [4]     AlgorithmConstraints OPTIONAL  -- TimeStamping Authority
             }

AlgorithmConstraints ::= SEQUENCE OF AlgAndLength

AlgAndLength ::= SEQUENCE {
    algID           OBJECT IDENTIFIER,
    minKeyLength    INTEGER     OPTIONAL, -- Minimum key length in bits
    other       SignPolExtensions OPTIONAL
         }

SignPolExtensions ::= SEQUENCE OF SignPolExtn

SignPolExtn ::= SEQUENCE {
        extnID      OBJECT IDENTIFIER,
        extnValue   OCTET STRING  }

END -- ETS-ElectronicSignaturePolicies-88syntax --
```

A.2 Signature policy definitions using X.680 (2002) ASN.1 syntax

NOTE: The ASN.1 module defined in clause A.1 has precedence over that defined in this clause using syntax defined in ITU-T Recommendation X.680 [16] in the case of any conflict.

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

DEFINITIONS EXPLICIT TAGS ::=
BEGIN
```

```
-- EXPORTS All -

IMPORTS


-- Internet X.509 Public Key Infrastructure - Certificate and CRL Profile: RFC 2459 or RFC 3280
    Certificate, AlgorithmIdentifier, CertificateList, Name, GeneralNames, GeneralName,
    DirectoryString, Attribute, AttributeTypeAndValue, AttributeType, AttributeValue,
    PolicyInformation

  FROM PKIX1Explicit93
    {iso(1) identified-organization(3) dod(6) internet(1)
    security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit-88(1)}
;
-- The structures may also be imported from :
-- PKIX1Explicit88 { iso(1) identified-organization(3) dod(6) internet(1)
-- security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit(18) }


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

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

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


-- Signature Policy Specification
-- ==============================

SignaturePolicy ::= SEQUENCE {
    signPolicyHashAlg      AlgorithmIdentifier,
    signPolicyInfo         SignPolicyInfo,
    signPolicyHash         SignPolicyHash     OPTIONAL }

SignPolicyHash ::= OCTET STRING

SignPolicyInfo ::= SEQUENCE {
    signPolicyIdentifier            SignPolicyId,
    dateOfIssue                     GeneralizedTime,
    policyIssuerName                PolicyIssuerName,
    fieldOfApplication              FieldOfApplication,
    signatureValidationPolicy       SignatureValidationPolicy,
    signPolExtensions               SignPolExtensions       OPTIONAL
    }

SignPolicyId ::= OBJECT IDENTIFIER

PolicyIssuerName ::= GeneralNames

FieldOfApplication ::= DirectoryString

SignatureValidationPolicy ::= SEQUENCE {
    signingPeriod          SigningPeriod,
    commonRules            CommonRules,
    commitmentRules        CommitmentRules,
    signPolExtensions      SignPolExtensions        OPTIONAL
     }

SigningPeriod ::= SEQUENCE {
    notBefore   GeneralizedTime,
    notAfter    GeneralizedTime OPTIONAL }

CommonRules  ::= SEQUENCE {
    signerAndVeriferRules          [0]  SignerAndVerifierRules     OPTIONAL,
    signingCertTrustCondition      [1]  SigningCertTrustCondition   OPTIONAL,
```

```
    timeStampTrustCondition        [2]  TimestampTrustCondition     OPTIONAL,
    attributeTrustCondition        [3]  AttributeTrustCondition     OPTIONAL,
    algorithmConstraintSet         [4]  AlgorithmConstraintSet      OPTIONAL,
    signPolExtensions              [5]  SignPolExtensions           OPTIONAL
    }

CommitmentRules ::= SEQUENCE OF CommitmentRule

CommitmentRule  ::= SEQUENCE {
    selCommitmentTypes                  SelectedCommitmentTypes,
    signerAndVeriferRules          [0]  SignerAndVerifierRules      OPTIONAL,
    signingCertTrustCondition      [1]  SigningCertTrustCondition   OPTIONAL,
    timeStampTrustCondition        [2]  TimestampTrustCondition     OPTIONAL,
    attributeTrustCondition        [3]  AttributeTrustCondition     OPTIONAL,
    algorithmConstraintSet         [4]  AlgorithmConstraintSet      OPTIONAL,
    signPolExtensions              [5]  SignPolExtensions           OPTIONAL
    }

SelectedCommitmentTypes ::= SEQUENCE OF CHOICE {
    empty                       NULL,
    recognizedCommitmentType    CommitmentType }

CommitmentType ::= SEQUENCE {
    identifier          CommitmentTypeIdentifier,
    fieldOfApplication  [0] FieldOfApplication OPTIONAL,
    semantics           [1] DirectoryString OPTIONAL }

SignerAndVerifierRules ::= SEQUENCE {
    signerRules      SignerRules,
    verifierRules    VerifierRules }

SignerRules ::= SEQUENCE {
    externalSignedData         BOOLEAN  OPTIONAL,
            -- True if signed data is external to CMS structure
            -- False if signed data part of CMS structure
            -- not present if either allowed
    mandatedSignedAttr         CMSAttrs,    -- Mandated CMS signed attributes
    mandatedUnsignedAttr       CMSAttrs,    -- Mandated CMS unsigned attributed
    mandatedCertificateRef     [0] CertRefReq DEFAULT signerOnly,
                -- Mandated Certificate Reference
    mandatedCertificateInfo    [1] CertInfoReq DEFAULT none,
                -- Mandated Certificate Info
    signPolExtensions        [2] SignPolExtensions      OPTIONAL
                        }

CMSAttrs ::= SEQUENCE OF OBJECT IDENTIFIER

CertRefReq ::= ENUMERATED {
                signerOnly (1),     -- Only reference to signer cert mandated
                fullPath (2)
                        -- References for full cert path up to a trust point required
                                    }

CertInfoReq ::= ENUMERATED {
                none (0)    ,           -- No mandatory requirements
                signerOnly (1)  ,       -- Only reference to signer cert mandated
                fullPath (2)
                        -- References for full cert path up to a trust point mandated
        }

VerifierRules ::= SEQUENCE {
        mandatedUnsignedAttr    MandatedUnsignedAttr,
        signPolExtensions       SignPolExtensions       OPTIONAL
        }

MandatedUnsignedAttr ::=  CMSAttrs    -- Mandated CMS unsigned attributed


CertificateTrustTrees ::=   SEQUENCE OF CertificateTrustPoint

CertificateTrustPoint ::= SEQUENCE {
    trustpoint              Certificate,                        -- self-signed certificate
    pathLenConstraint   [0] PathLenConstraint   OPTIONAL,
    acceptablePolicySet [1] AcceptablePolicySet OPTIONAL,  -- If not present "any policy"
    nameConstraints     [2] NameConstraints     OPTIONAL,
    policyConstraints   [3] PolicyConstraints   OPTIONAL }
```

```
PathLenConstraint    ::=   INTEGER (0..MAX)

AcceptablePolicySet ::= SEQUENCE OF CertPolicyId

CertPolicyId ::= OBJECT IDENTIFIER

NameConstraints ::= SEQUENCE {
          permittedSubtrees       [0]     GeneralSubtrees OPTIONAL,
          excludedSubtrees        [1]     GeneralSubtrees OPTIONAL }

      GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree

      GeneralSubtree ::= SEQUENCE {
           base                    GeneralName,
           minimum         [0]     BaseDistance DEFAULT 0,
           maximum         [1]     BaseDistance OPTIONAL }

      BaseDistance ::= INTEGER (0..MAX)

PolicyConstraints ::= SEQUENCE {
        requireExplicitPolicy           [0] SkipCerts OPTIONAL,
        inhibitPolicyMapping            [1] SkipCerts OPTIONAL }

SkipCerts ::= INTEGER (0..MAX)

CertRevReq ::= SEQUENCE {
    endCertRevReq   RevReq,
    caCerts         [0] RevReq
      }

RevReq ::= SEQUENCE  {
    enuRevReq  EnuRevReq,
    exRevReq    SignPolExtensions OPTIONAL}

EnuRevReq  ::= ENUMERATED {
    clrCheck    (0), --Checks shall be made against current CRLs
            -- (or authority revocation lists)
    ocspCheck   (1), -- The revocation status shall be checked
            -- using the Online Certificate Status Protocol (RFC 2450)
    bothCheck   (2),    -- Both CRL and OCSP checks shall be carried out
    eitherCheck (3),    -- At least one of CRL or OCSP checks shall be carried out
    noCheck     (4),    -- no check is mandated
    other       (5)     -- Other mechanism as defined by signature policy extension -- }

SigningCertTrustCondition ::=   SEQUENCE {
     signerTrustTrees              CertificateTrustTrees,
     signerRevReq                  CertRevReq
}

TimestampTrustCondition ::= SEQUENCE {
    ttsCertificateTrustTrees    [0]     CertificateTrustTrees    OPTIONAL,
    ttsRevReq                   [1]     CertRevReq              OPTIONAL,
    ttsNameConstraints          [2]     NameConstraints         OPTIONAL,
    cautionPeriod               [3]     DeltaTime               OPTIONAL,
    signatureTimestampDelay     [4]     DeltaTime               OPTIONAL }

DeltaTime ::= SEQUENCE {
    deltaSeconds    INTEGER,
    deltaMinutes    INTEGER,
    deltaHours      INTEGER,
    deltaDays       INTEGER }

AttributeTrustCondition ::= SEQUENCE {
    attributeMandated            BOOLEAN,             -- Attribute shall be present
    howCertAttribute             HowCertAttribute,
    attrCertificateTrustTrees   [0] CertificateTrustTrees   OPTIONAL,
    attrRevReq                  [1] CertRevReq             OPTIONAL,
    attributeConstraints        [2] AttributeConstraints   OPTIONAL }


HowCertAttribute ::= ENUMERATED {
    claimedAttribute    (0),
    certifiedAttribtes  (1),
    either              (2) }
```

```
AttributeConstraints ::= SEQUENCE {
    attributeTypeConstraints    [0] AttributeTypeConstraints  OPTIONAL,
    attributeValueConstraints   [1] AttributeValueConstraints OPTIONAL }


AttributeTypeConstraints ::= SEQUENCE OF AttributeType

AttributeValueConstraints ::= SEQUENCE OF AttributeTypeAndValue


AlgorithmConstraintSet ::= SEQUENCE {   -- Algorithm constraints on:
signerAlgorithmConstraints  [0]     AlgorithmConstraints OPTIONAL, -- signer
eeCertAlgorithmConstraints  [1]     AlgorithmConstraints OPTIONAL, -- issuer of end entity certs
caCertAlgorithmConstraints  [2]     AlgorithmConstraints OPTIONAL, -- issuer of CA certificates
aaCertAlgorithmConstraints  [3]     AlgorithmConstraints OPTIONAL, -- Attribute Authority
tsaCertAlgorithmConstraints [4]     AlgorithmConstraints OPTIONAL  -- TimeStamping Authority -- }

AlgorithmConstraints ::= SEQUENCE OF AlgAndLength

AlgAndLength ::= SEQUENCE {
    algID           OBJECT IDENTIFIER,
    minKeyLength    INTEGER     OPTIONAL, -- Minimum key length in bits
    other       SignPolExtensions OPTIONAL }

SignPolExtensions ::= SEQUENCE OF SignPolExtn

SignPolExtn ::= SEQUENCE {
        extnID      OBJECT IDENTIFIER,
        extnValue   OCTET STRING  }

END -- ETS- ElectronicSignaturePolicies-97Syntax
```

Annex B:
What is a signature policy and signature validation policy

B.0 Introduction

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

Electronic signatures are commonly applied within the context of a legal or contractual framework. This establishes the requirements on the electronic signatures and any special semantics (e.g. agreement, intent). These requirements may be defined in very general abstract terms or in terms of detailed rules. The specific semantics associated with an electronic signature implied by a legal or contractual framework are outside the scope of the present document.

If the signature policy is recognized, within the legal/contractual context, as providing commitment, then the signer explicitly agrees with terms and conditions which are implicitly or explicitly part of the signed data.

When two independent parties want to evaluate an electronic signature, it is fundamental that they get the same result. It is therefore important that the conditions agreed by the signer at the time of signing are indicated to the verifier and any arbitrator. An aspect that enables this to be known by all parties is the signature policy. The technical implications of the signature policy on the electronic signature with all the validation data are called the "Signature Validation Policy". The signature validation policy specifies the rules used to validate the signature.

A signature policy may be explicitly identifier or may be implied by the semantics of the data being signed and other external data. TS 101 733 (see bibliography) does not mandate the form and encoding of the specification of the signature policy. However, for a given signature policy there shall be one definitive form and an explicit policy must have a unique binary encoded value.

The present document includes the formal structure for an explicit signature validation policy based on the use of the ASN.1 syntax.

Given the specification of the explicit signature policy and its hash value an implementation of a verification process shall obey the rules defined in the specification.

TS 101 733 places no restriction on how a signature policy should be implemented. Provided that the implementation allows to support the conformance requirements defined for the electronic signature formats in TS 101 733 V1.5.1 clause 8.1 for a BES and clause 8.2 for EPES. This includes:

- A validation process that supports a specific signature policy as identified by the signature policy OID. Such an implementation should conform to a human readable description provided all the processing rules of the signature policy are clearly defined. However, if additional policies need to be supported, then such an implementation would need to be customized for each additional policy. This type of implementation may be simpler to implement initially, but can be difficult to enhance to support numerous additional signature policies.

- A validation process that is dynamically programmable and able to adapt its validation rules in accordance with a description of the signature policy provided in a computer-processable language. The present document defines such a policy using an ASN.1 structure (see clause 6.1). This type of implementation could support multiple signature policies without being modified every time, provided all the validation rules specified as part of the signature policy are known by the implementation. (i.e. only requires modification if there are additional rules specified).

The precise content of a signature policy is not mandated by TS 101 733. However, a signature policy shall be sufficiently definitive to avoid any ambiguity as to its implementation requirements. It shall be absolutely clear under which conditions an electronic signature should be accepted. For this reason, it should contain the following information:

- General information about the signature policy which includes:
  - a unique identifier of the policy;
  - the name of the issuer of the policy;
  - the date the policy was issued;
  - the field of application of the policy.
- The signature verification policy which includes:
  - the signing period;
  - a list of recognized commitment types;
  - rules for Use of Certification Authorities;
  - rules for Use of Revocation Status Information;
  - rules for Use of Roles;
  - rules for use of Time-stamping and Time-marking;
  - signature verification data to be provided by the signer/collected by verifier;
  - any constraints on signature algorithms and key lengths.
- Other signature policy rules required to meet the objectives of the signature.

Variations of the validation policy rules may apply to different commitment types.

B.1 Identification of signature policy

When data is signed the signer indicates that a signature policy applies to that electronic signature by including a signed attribute which specifies either that an explicit or an implicit signature policy is applicable. The signer and verifier shall apply the rules specified by the identified policy when validating the signature. If the signature policy is explicit the signer shall include the hash of the signature policy, so it can be verified that the policy selected by the signer is identical to the one being used the verifier.

A signature policy may be qualified by additional information. This may include:

- a URL where a copy of the Signature Policy may be obtained;
- a user notice that should be displayed when the signature is verified.

If the signature policy attribute is absent and no signature policy is identified then the signature may be assumed to have been generated/verified without any policy constraints, and hence may be given no specific legal or contractual significance through the context of a signature policy.

An explicit "Signature Policy" will be identifiable by an OID (Object Identifier) and verifiable using a hash of the signature policy.

B.2 General signature policy information

General information should be recorded about the signature policy along with the definition of the rules which form the signature policy as described in subsequent clauses. This should include:

- **Policy Object Identifier:** the "Signature Policy" will be identifiable by an OID (Object Identifier) whose last component (i.e. right most) is an integer that is specific to a particular version issued on the given date.
- **Date of issue:** when the "Signature Policy" was issued.
- **Signature Policy Issuer name:** an identifier for the body responsible for issuing the Signature Policy. This may be used by the signer or verifier in deciding if a policy is to be trusted, in which case the signer/verifier shall authenticate the origin of the signature policy as coming from the identified issuer.
- **Signing period:** the start time and date, optionally with an end time and date, for the period over which the signature policy may be used to generate electronic signatures.
- **Field of application:** this defines in general terms the general legal/contract/application contexts in which the signature policy is to be used and the specific purposes for which the electronic signature is to be applied.

B.3 Recognized commitment types

The signature validation policy may recognize one or more types of commitment as being supported by electronic signatures produced under the security policy.

If an electronic signature does not contain a recognized commitment type then the semantics of the electronic signature is dependent on the data being signed and the context in which it is being used.

Only recognized commitment types are allowed in an electronic signature.

The definition of a commitment type includes:

- the object identifier for the commitment;
- a qualifier.

The qualifier provides more information about the commitment, for example it could provide:

- information about the context be it contractual/legal/application specific.

The definition of a commitment type can be registered:

- as part of the validation policy;
- as part of the application/contract/legal environment;
- as part of generic register of definitions.

The legal/contractual context will determine the rules applied to the signature, as defined by the signature policy and its recognized commitment types, make it fit for purpose intended.

B.4 Rules for use of certification authorities

The certificate validation process of the verifier, and hence the certificates that may be used by the signer for a valid electronic signature, may be constrained by the combination of the trust point and certificate path constraints in the signature validation policy.

B.4.1 Trust points

The signature validation policy defines the certification authority trust points that are to be used for signature verification. Several trust points may be specified under one signature policy. Specific trust points may be specified for a particular type of commitment defined under the signature policy. For a signature to be valid a certification path shall exists between the Certification Authority that has granted the certificate selected by the signer (i.e. the used user-certificate) and one of the trust point of the "Signature Validation Policy".

B.4.2 Certification path

There may be constraints on the use of certificates issued by one or more CA(s) in the certificate chain and trust points. The two prime constraints are certificate policy constraints and naming constraints.

- Certificate policy constraints limit the certification chain between the user certificate and the certificate of the trusted point to a given set of certificate policies, or equivalents identified through certificate policy mapping.
- The naming constraints limit the forms of names that the CA is allowed to certify.

Name constraints are particularly important when a "Signature policy" identifies more than one trust point. In this case, a certificate of a particular trusted point may only be used to verify signatures from users with names permitted under the name constraint.

Certificate Authorities may be organized in a tree structure, this tree structure may represent the trust relationship between various CA(s) and the users CA. Alternatively, a mesh relationship may exist where a combination of tree and peer cross-certificates may be used. The requirement of the certificate path in the present document is that it provides the trust relationship between all the CAs and the signers user certificate. The starting point from a verification point of view, is the "trust point". A trust point, usually a CA that publishes self-certified certificates, is the starting point from which the verifier verifies the certificate chain. Naming constraints may apply from the trust point, in which case they apply throughout the set of certificates that make up the certificate path down to the signer's user certificate.

Policy constraints can be easier to process but to be effective require the presence of a certificate policy identifier in the certificates used in a certification path.

Certificate path processing, thus generally starts with one of the trust point from the signature policy and ends with the user certificate.

The certificate path processing procedures defined in RFC 2459 [6], clause 6 identifies the following initial parameters that are selected by the verifier in certificate path processing:

- acceptable certificate policies;
- naming constraints in terms of constrained and excluded naming subtree;
- requirements for explicit certificate policy indication and whether certificate policy mapping are allowed;
- restrictions on the certificate path length.

The signature validation policy identifies constraints on these parameters.

B.5 Rules for the use of time-stamping and time-marking

In order for a digital signature to be valid, it must be proven that the digital signature was applied while the signer's certificate was valid.

A timestamp or time mark applied to a digital signature value proves that the digital signature was created before the date included in the time-stamp or time mark.

To prove the digital signature was generated while the signer's certificate was valid, the digital signature must be verified and the following conditions satisfied:

1) the time-stamp or time mark must be applied before the end of the validity period of the signer's certificate;

2) the time-stamp or time mark must be applied either while the signer's certificate was not revoked or before the revocation date of the certificate.

Thus a time-stamp or time mark applied in this manner proves that the digital signature was created while the signer's certificate was valid, this concept can be extended to prove the validity of a digital signature over the whole or any certificate chain.

There will necessarily be some delay between the time that a signature is created and the time the signer's digital signature is time-stamped or time marked. The longer this elapsed period the greater the risk of the signature being invalidated due to compromise or deliberate revocation of its private signing key by the signer. Since a certificate can be in practice revoked at any time, it is the interest from the verifier to get a time-stamp token as soon as possible.

The following rules should be used when specifying, constraints on the certificate paths for time-stamping authorities, constraints on the time-stamping authority names and general timing constraints.

B.5.1 Trust points and certificate paths

Signature keys from time-stamping authorities will need to be supported by a certification path. The certification path used for time-stamping authorities requires a trust point and possibly path constraints in the same way that the certificate path for the signer's key.

B.5.2 Time-stamping authority names

Restrictions may need to be placed by the validation policy on the named entities that may act as time-stamping authorities.

B.5.3 Timing constraints - cautionary period

Before an electronic signature may really be valid, the verifier has to be sure that the holder of the private key was really the only one in possession of key at the time of time-stamping. However, there is an inevitable delay between a compromise or loss of key being noted, and a report of revocation being distributed. To allow greater confidence in the validity of a signature, a "cautionary period" may be mandated before a signature may be said to be valid. A verifier shall wait until the termination of the cautionary period to check the revocation status of the certification path. The validation policy may specify such a cautionary period.

B.5.4 Timing constraints - time-stamp delay

In order to get greater confidence of the claimed signing time as indicated by the signer, the signature policy should specify a maximum acceptable delay between the signing time as claimed by the signer and the time included within the time-stamp token.

B.6 Revocation rules

The signature policy should define rules specifying requirements for the use of certificate revocation lists (CRLs) and/or on-line certificate status check service to check the validity of a certificate. These rules specify the mandated minimum checks that shall be carried out. These checks shall be carried out once the cautionary period is over.

It is expected that in many cases either check may be selected with checks of CRLs being carried out for certificate status that are unavailable from OCSP servers. The verifier may take into account information in the certificate in deciding how best to check the revocation status (e.g. a certificate extension field about authority information access or a CRL distribution point) provided that it does not conflict with the signature policy revocation rules.

B.7 Rules for the use of roles

Roles can be supported as claimed roles or as certified roles using Attribute Certificates.

B.7.1 Attribute values

When signature under a role is mandated by the signature policy, then either Attribute Certificates may be used or the signer may provide a claimed role attribute. The acceptable attribute types or values may be dependent on the type of commitment. For example, a user may have several roles that allow the user to sign data that imply commitments based on one or more of his roles.

B.7.2 Trust points for certified attributes

When a signature under a certified role is mandated by the signature policy, Attribute Authorities are used and need to be validated as part of the overall validation of the electronic signature. The trust points for Attribute Authorities do not need to be the same as the trust points to evaluate a certificate from the CA of the signer. Thus the trust point for verifying roles need not be the same as trust point used to validate the certificate path of the user's key.

Naming and certification policy constraints may apply to the AA in similar circumstance to when they apply to CA. Constraints on the AA and CA need not be exactly the same.

AA(s) may be used when a signer is creating a signature on behalf of an organization, they can be particularly useful when the signature represents an organizational role. AA(s) may or may not be the same authority as CA(s).

Thus, the Signature Policy identifies trust points that can be used for Attribute Authorities, either by reference to the same trust points as used for Certification Authorities, or by an independent list.

B.7.3 Certification path for certified attributes

Attribute Authorities may be organized in a tree structure in similar way to CAs, where the AAs are the leaves of such a tree. Naming and other constraints may be required on attribute certificate paths in a similar manner to other electronic signature certificate paths.

Thus, the Signature Policy identifies constraints on the following parameters used as input to the certificate path processing:

- acceptable certificate policies, including requirements for explicit certificate policy indication and whether certificate policy mapping is allowed;
- naming constraints in terms of constrained and excluded naming subtrees;
- restrictions on the certificate path length.

B.8 Rules for verification data to be followed

By specifying the requirements on the signer and verifier the responsibilities of the two parties can be clearly defined to establish all the necessary information.

These verification data rules should include:

- requirements on the signer to provide given signed attributes;
- requirements on the verifier to obtain additional certificates, CRLs, results of on line certificate status checks and to use timestamps (if no already provided by the signer).

B.9 Rules for algorithm constraints and key lengths

The signature validation policy may identify a set of signing algorithms (hashing, public key, combinations) and minimum key lengths that may be used:

- by the signer in creating the signature;
- in end entity public key Certificates;
- CA Certificates;
- attribute Certificates;
- by the time-stamping authority.

B.10 Other signature policy rules

The signature policy may specify additional policy rules, for example rules that relate to the environment used by the signer. These additional rules may be defined in computer processable and/or human readable form.

B.11 Signature policy protection

When signer or verifier obtains a copy of the Signature Policy from an issuer, the source should be authenticated (for example by using electronic signatures).

When the signer references a signature policy the Object Identifier (OID) of the policy, the hash value and the hash algorithm OID of that policy shall be included in the Electronic Signature.

It is a mandatory requirement of both ETSI TS 101 733 and the present document that the signature policy value computes to one, and only one hash value using the specified hash algorithm. This means that there shall be a single binary value of the encoded form of the signature policy for the unique hash value to be calculated. For example, there may exist a particular file type, length and format on which the hash value is calculated which is fixed and definitive for a particular signature policy.

The hash value may be obtained by:

- the signer performing his own computation of the hash over the signature policy using his preferred hash algorithm permitted by the signature policy, and the definitive binary encoded form;
- the signer, having verified the source of the policy, may use both the hash algorithm and the hash value included in the computer processable form of the policy (see clause B.1).

Annex C: Bibliography

- Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.
- ETSI TS 101 862: "Qualified certificate profile".
- ETSI TS 101 456: "Policy requirements for certification authorities issuing qualified certificates".
- ETSI TS 101 861: "Time stamping profile".
- ETSI TS 102 023: "Electronic Signatures and Infrastructures (ESI); Policy requirements for time-stamping authorities".
- ETSI TS 101 903: "XML Advanced Electronic Signatures (XAdES)".
- ETSI TS 101 733 (V1.4.0): "Electronic Signatures and Infrastructures (ESI); Electronic Signature Formats".
- ETSI TR 102 038: " TC Security - Electronic Signatures and Infrastructures (ESI); XML format for signature policies".
- ETSI TR 102 045: " Electronic Signatures and Infrastructures (ESI); Signature policy for extended business model".
- ITU-T Recommendation X.209 (1988): "Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1)".
- ITU-T Recommendation X.681 (1997) | ISO/IEC 8824-2: "Information technology - Abstract Syntax Notation One (ASN.1): Information object specification".
- ISO/IEC 9796 (1991): "Information technology - Security techniques - Digital signature scheme giving message recovery".
- ISO/IEC 9796-2 (1997): "Information technology - Security techniques - Digital signature schemes giving message recovery - Part 2: Mechanisms using a hash-function".
- ISO/IEC CD 9796-4: "Digital signature schemes giving message recovery - Part 4: Discrete logarithm based mechanisms".
- ISO/IEC 10118-1 (1994): "Information technology - Security techniques - Hash-functions - Part 1: General".
- ISO/IEC 10118-2 (1994): "Information technology - Security techniques - Hash-functions - Part 2: Hash-functions using an n-bit block cipher algorithm".
- ISO/IEC 10118-3 (1997): "Information technology - Security techniques - Hash-functions - Part 3: Dedicated hash-functions".
- ISO/IEC FCD 10118-4: "Information technology - Security techniques - Hash-functions - Part 4: Hash-functions using modular arithmetic".
- ISO/IEC FCD 14888-1: "Digital signatures with appendix - Part 1: General".
- ISO/IEC FCD 14888-2: "Digital signatures with appendix - Part 2: Identity-based mechanisms".
- ISO/IEC FCD 14888-3: "Digital signatures with appendix - Part 3: Certificate-based mechanisms".
- ISO/IEC WD 15946-2: "Cryptographic techniques based on elliptic curves - Part 2: Digital signatures".
- IETF RFC 2527 (1999): "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework".

- IETF RFC 2528 (1999): "Internet X.509 Public Key Infrastructure; Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates".
- IETF RFC 1320 (PS 1992): "The MD4 Message-Digest Algorithm".
- IETF RFC 1321: "The MD5 Message-Digest Algorithm".
- IETF RFC 2068: "Hypertext Transfer Protocol - HTTP/1.1".
- IETF RFC 2246: "The TLS Protocol Version 1.0".
- IETF RFC 2313 (1998): "PKCS 1: RSA Encryption Version, Version 1.5".
- IETF RFC 2437: "PKCS #1: RSA Cryptography Specifications Version 2.0".
- IETF RFC 2479: "Independent Data Unit Protection Generic Security Service Application Program Interface (IDUP-GSS-API)".
- IETF RFC 2585 (1999): "Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP".
- PKCS #1 V2.0 (1998): "RSA Cryptography Standard", RSA Laboratories.
- IEEE P1363: "Standard Specifications for Public-Key Cryptography".
- FIPS Publication 180-1 (1995): "Secure Hash Standard".
- FIPS Publication 186 (1994): "Digital Signature Standard".
- ANS X9.30-1 (1997): "Public Key Cryptography for the Financial Services Industry - Part 1: The Digital Signature Algorithm (DSA)".
- ANS X9.30-2 (1997): "Public Key Cryptography for the Financial Services Industry - Part 2: The Secure Hash Algorithm (SHA-1)".
- ANS X9.31-1: "Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry - Part 1: The RSA Signature Algorithm".
- ANS X9.31-2: "Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry - Part 2: Hash Algorithms".
- ANS X9.62 (draft): "Public Key Cryptography for the Financial Services Industry - The Elliptic Curve Digital Signature Algorithm (ECDSA)".
- ITU-T Recommendation X.520 (1997): "Information technology - Open Systems Interconnection - The Directory: Selected attribute types".
- IETF RFC 2743: "Generic Security Service Application Program Interface Version 2, Update 1".

The following documents are IETF Internet-Drafts and working documents of the IETF. For up to date information reference should be to the latest versions. Also later versions of the documents may make the referenced documents below obsolete:

- Certificate Management Messages over CMS;
- Internet X.509 Public Key Infrastructure Representation of Elliptic Curve Digital Signature Algorithm (ECDSA) Keys and Signatures in Internet X.509 Public Key Infrastructure Certificates;
- Time Stamp Protocol (TPS).
- Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols.
- Internet X.509 Public Key Infrastructure PKIX Roadmap.
- Internet X.509 Public Key Infrastructure Qualified Certificates.
- Diffie-Hellman Proof-of-Possession Algorithms.

- An Internet Attribute Certificate Profile for Authorization.
- Basic Event Representation Token v1.
- Internet X.509 Public Key Infrastructure Extending trust in non-repudiation tokens in time.
- Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv3.
- Simple Certificate Validation Protocol (SCVP).
- Using HTTP as a Transport Protocol for CMP.
- Using TCP as a Transport Protocol for CMP.
- Limited Attribute Certificate Acquisition Protocol.
- OCSP Extensions.
- Certificate and CRL Profile.
- A String Representation of General Name.
- XML-Signature Requirements.
- XML-Signature Core Syntax.
- Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures.
- W3C Recommendation: "XML Schema Part 1: Structures".
- W3C Recommendation: "XML Schema Part 2: Datatypes".
- IETF RFC 2634 (June 1999): "Enhanced Security Services for S/MIME".
- ITU-T Recommendation X.509: "Information technology - Open Systems Interconnection - The directory: Public-key and attribute certificate frameworks".
- ETSI TS 101 861: "Time stamping profile".

History

Document history		
V1.1.1	December 2003	Publication