

## PKI applications (C2)

### Protocols and standards

Pascal Steichen (MSSI-uni.lu) - 31/03/2007 (03)

- 1. Internet X.509 Public Key Infrastructure (PKIX)
  - 1.1. PKI - Public-Key Infrastructure
  - 1.2. PMI - Privilege Management Infrastructure
- 2. Certificate and Certificate Revocation List (CRL) Profile - RFC 3280
  - 2.1. Certificate
  - 2.2. CRL (Certificate Revocation List)
  - 2.3. Certification Path Validation
- 3. Online Certificate Status Protocol (OCSP) - RFC 2560
  - 3.1. Request Syntax
  - 3.2. OCSP Response
- 4. Certificate Policy and Certification Practices Framework - RFC 3647
  - 4.1. Certificate policy (CP)
  - 4.2. Certification Practice Statement (CPS)
  - 4.3. Recommended CP or CPS outline
- 5. Time-Stamping Authorities (TSAs) - RFC 3628
  - 5.1. Time-Stamp Token
  - 5.2. Clock Synchronization with UTC
  - 5.3. Time-Stamp Protocol (TSP) - RFC 3161
  - 5.4. Request Format
  - 5.5. Response Format
- 6. Public-Key Cryptography Standards (PKCS)
- 7. Bibliographic references

## 1. Internet X.509 Public Key Infrastructure (PKIX)

PKIX standardisation areas:

1. Profiles of X.509 Public Key Certificates and X.509 Certificate Revocation Lists (CRLs).

It describes the basic certificate fields and the extensions to be supported for the Certificates and the Certificate Revocation Lists. Then, it talks about the basic and extended Certificate Path Validation. Finally, it covers the supported cryptographic algorithms.

2. Management protocols.

Management protocols are the protocols that are required to support on-line interactions between PKI user and management entities. The possible set of functions that can be supported by management protocols is

- registration of entity, that takes place prior to issuing the certificate,
- initialisation, for example generation of key-pair,
- certification, the issuance of the certificate,
- key-pair recovery, the ability to recover lost keys,
- key-pair update, when the certificate expires and a new key-pair and certificate have to be generated,
- revocation request, when an authorised person advises the CA to include a specific certificate into the revocation list,
- cross-certification, when two CAs exchange information in order to generate a cross-certificate.

The PKIX standard first discusses the assumptions and restrictions of the protocols. Then, it provides the data structures used for the PKI management messages and defines the functions that conforming implementations must carry out. Finally, it describes a simple protocol for transporting PKI messages.

3. Operational protocols.

The operational protocols are the protocols that are required to deliver certificates and CRLs (or status information) to certificate-using client systems. There is an emphasis to have a variety of distribution mechanisms for the certificates and the CRLs, using for example, LDAP, HTTP and FTP. For example, the retrieval of the CRL by a merchant to check whether a certificate is valid, constitutes an operational protocol.

Currently they describe how LDAP, FTP and HTTP can be used as operational protocols, as well how online validation (OCSP) should be implemented.

4. Certificate policies and Certificate Practice Statements.

The Certificate Policies and the Certificate Practice Statements are recommendations of documents that will describe the obligations and other rules with regard the usage of the Certificate.

The purpose of this document is to establish a clear relationship between certificate policies and CPSs, and to present a framework to assist the writers of certificate policies or CPSs with their tasks. In particular, the framework identifies the elements that may need to be considered in formulating a certificate policy or a CPS.

The purpose is not to define particular certificate policies or CPSs, per se.

5. Time-stamping and data-certification/validation services.

The time-stamping services define a trusted third-party that creates time stamp tokens in order to indicate that a datum existed at a particular point in time. The data certification and validation services provide certification of possession of data and claim of possession of data, and validation of digitally signed documents and certificates.

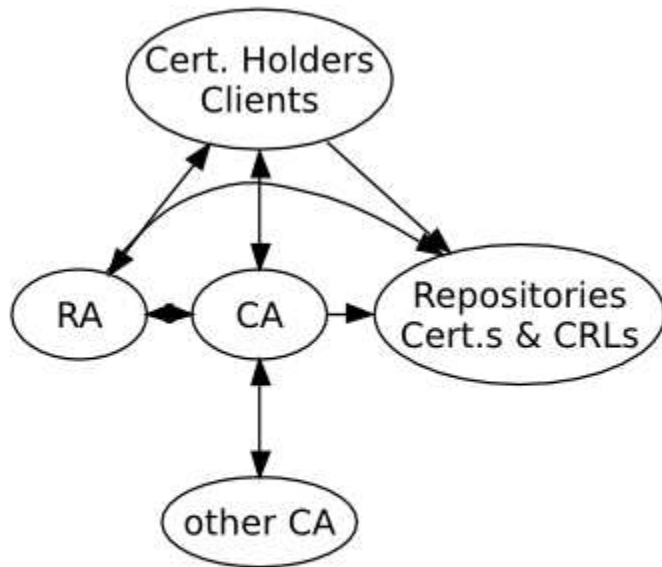
The relevant Request For Comments (RFC) documents are depicted in the following table

Subject	RFC
Certificate and Certificate Revocation List (CRL) Profile	RFC 3280
Certificate Management Protocol (CMP)	RFC 4210
Operational protocols	RFC 3494 (LDAP), RFC 2585, RFC 2560 (OCSP)
Certificate Policy and Certification Practices Framework	RFC 3647
Time-stamping and data-certification services	RFC 3628

### 1.1. PKI - Public-Key Infrastructure

A PKI is a set of hardware, software, people, policies and procedures needed to create, manage, store, distribute and revoke PKCs based on public-key cryptography.

A PKI consists of five types of components.



Type of component	Description
Certification Authorities (CAs)	to issue and revoke PKCs
Registration Authorities (RAs)	to vouch for the binding between public keys and certificate holder identities and other attributes
Certificate holders (end entities, EE)	to sign and encrypt digital documents
Clients (end entities, EE)	to validate digital signatures and their certification path from a known public key of a trusted CA
Repositories	to store and make available certificates and Certificate Revocation Lists (CRLs)

The End-entity, using management transactions, sends its certificate request to the Registration Authority for approval. If it is actually approved, it is forwarded to the Certification Authority for signing. The

Certification Authority verifies the certificate request and if it passes the verification, it is signed and the Certificate is produced. To public the Certificate, the CA sends it to Certificate Repository for collection from the End-entity.

The diagram shows that the End-entity can communicate directly with the CA. According to the PKIX recommendations, it is possible to implement the functionality within the CA. Although it is a bit confusing, the diagram shows all possible communications, regardless of the implementation decisions.

Additionally, both the CA and RA are shown to deliver Certificates to the repository. Depending on the implementation, one of the two is chosen.

For the issue of the revocation of the certificates, a similar course with the generation of the Certificates is taken. The End-entity asks the RA to have its Certificate revoked, the RA decides and possibly forwards it to the CA, the CA updates the revocation list and publishes it on the CRL repository.

Finally, the End-entities can check the validity of a specific Certificate using an operational protocol.

- MUST This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- MUST NOT This phrase, or the phrase "SHALL NOT", mean that the

definition is an absolute prohibition of the specification.

**SHOULD** This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

**SHOULD NOT** This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

**MAY** This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option **MUST** be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option **MUST** be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

- PKI Management Requirements. The PKI management protocols ...

1. must conform to the ISO/IEC 9594-8/ITU-T X.509 standards;

*ISO/IEC 954-8:2005 Information technology -- Open Systems Interconnection -- The Directory: Public-key and attribute certificate frameworks*

2. must be possible to regularly update key pairs;

without affecting other key pairs;

3. use of confidentiality in PKI management protocols must be kept to a minimum

in order to ease acceptance in environments where strong confidentiality might cause regulatory problems;

4. must allow the use of different industry-standard cryptographic algorithms

(specifically including RSA, DSA, MD5, and SHA-1). This means that any given CA, RA, or end entity may, in principle, use whichever algorithms suit it for its own key pair(s);

5. must not preclude the generation of key pairs by the end-entity concerned, by an RA, or by a CA.

Key generation may also occur elsewhere, but for the purposes of PKI management we can regard key generation as occurring wherever the key is first present at an end entity, RA, or CA;

6. must support the publication of certificates by the end-entity concerned, by an RA, or by a CA.

Different implementations and different environments may choose any of the above approaches;

7. must support the production of Certificate Revocation Lists (CRLs)

by allowing certified end entities to make requests for the revocation of certificates. This must be done in such a way that the denial-of-service attacks, which are possible, are not made simpler.

8. must be usable over a variety of "transport" mechanisms,

specifically including mail, http, TCP/IP and ftp;

9. authority for certification creation rests with the CA.

No RA or end-entity equipment can assume that any certificate issued by a CA will contain what was requested; a CA may alter certificate field values or may add, delete, or alter extensions according to its operating policy. In other words, all PKI entities (end-entities, RAs, and CAs) must be capable of handling responses to requests for certificates in which the actual certificate issued is different from that requested (for example, a CA may shorten the validity period requested). Note that policy may dictate that the CA must not publish or otherwise distribute the certificate until the requesting entity has reviewed and accepted the newly-created certificate.

10. A graceful, scheduled change-over from one non-compromised CA key pair to the next (CA key update) must be supported

(note that if the CA key is compromised, re-initialization must be performed for all entities in the domain of that CA). An end entity whose PSE contains the new CA public key (following a CA key update) must also be able to verify certificates verifiable using the old public key. End entities who directly trust the old CA key pair must also be able to verify certificates signed using the new CA private key (required for situations where the old CA public key is "hardwired" into the end entity's cryptographic equipment).



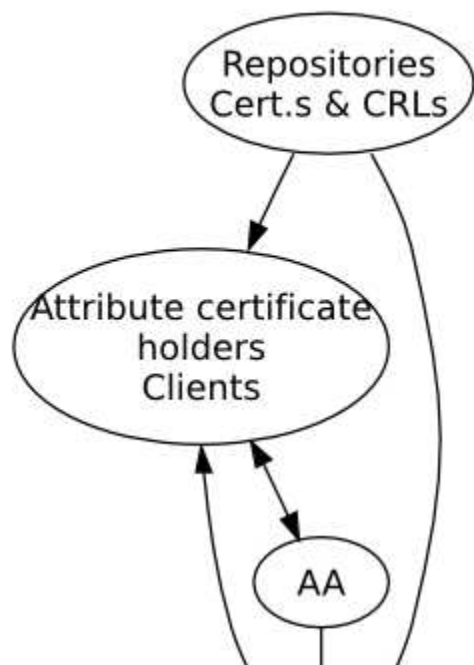
3. Certification: various operations result in the creation of new certificates:
  1. initial registration/certification: This is the process whereby an end entity first makes itself known to a CA or RA, prior to the CA issuing a certificate or certificates for that end entity. The end result of this process (when it is successful) is that a CA issues a certificate for an end entity's public key, and returns that certificate to the end entity and/or posts that certificate in a public repository. This process may, and typically will, involve multiple "steps", possibly including an initialization of the end entity's equipment. For example, the end entity's equipment must be securely initialized with the public key of a CA, to be used in validating certificate paths. Furthermore, an end entity typically needs to be initialized with its own key pair(s).
  2. key pair update: Every key pair needs to be updated regularly (i.e., replaced with a new key pair), and a new certificate needs to be issued.
  3. certificate update: As certificates expire, they may be "refreshed" if nothing relevant in the environment has changed.
  4. CA key pair update: As with end entities, CA key pairs need to be updated regularly; however, different mechanisms are required.
  5. cross-certification request: One CA requests issuance of a cross-certificate from another CA. For the purposes of this standard, the following terms are defined. A "cross-certificate" is a certificate in which the subject CA and the issuer CA are distinct and SubjectPublicKeyInfo contains a verification key (i.e., the certificate has been issued for the subject CA's signing key pair). When it is necessary to distinguish more finely, the following terms may be used: a cross-certificate is called an "inter-domain cross-certificate" if the subject and issuer CAs belong to different administrative domains; it is called an "intra-domain cross-certificate" otherwise.
  6. cross-certificate update: Similar to a normal certificate update, but involving a cross-certificate.
4. Certificate/CRL discovery operations: some PKI management operations result in the publication of certificates or CRLs:
  1. certificate publication: Having gone to the trouble of producing a certificate, some means for publishing it is needed. The "means" defined in PKIX MAY involve methods (LDAP, for example) as described in [RFC2559], [RFC2585] (the "Operational Protocols" documents of the PKIX series of specifications).
  2. CRL publication: As for certificate publication.
5. Recovery operations: some PKI management operations are used when an end entity has "lost" its PSE:
  - o key pair recovery: As an option, user client key materials (e.g., a user's private key used for decryption purposes) MAY be backed up by a CA, an RA, or a key backup system associated with a CA or RA. If an entity needs to recover these backed up key materials (e.g., as a result of a forgotten password or a lost key chain file), a protocol exchange may be needed to support such recovery.
6. Revocation operations: some PKI operations result in the creation of new CRL entries and/or new CRLs:
  - o revocation request: An authorized person advises a CA of an abnormal situation requiring certificate revocation.

Note that on-line protocols are not the only way of implementing the above operations. For all operations, there are off-line methods of achieving the same result, and this specification does not mandate use of on-line protocols. For example, when hardware tokens are used, many of the operations MAY be achieved as part of the physical token delivery.

## 1.2. PMI - Privilege Management Infrastructure

PMI is the set of hardware, software, people, policies and procedures needed to create, manage, store, distribute and revoke Attribute Certificates.

A PMI consists of five types of components.



Type of component	Description
Attribute Authorities (AAs)	to issue and revoke ACs (also called Attribute Certificate Issuer)
Attribute certificate verifier	to check the validity of an AC and then make use of the result
Attribute certificate holders (end entities, EEs)	to parse or process an AC
Clients (end entities, EEs)	to request an action for which authorisation checks are to be made
Repositories	to store and make available certificates and Certificate Revocation Lists (CRLs)

## 2. Certificate and Certificate Revocation List (CRL) Profile - RFC 3280

### 2.1. Certificate



The X.509 v3 certificate basic syntax is as follows. For signature calculation, the data that is to be signed is encoded using the ASN.1 distinguished encoding rules (DER) [X.690]. ASN.1 DER encoding is a tag, length, value encoding system for each element.

The Certificate is a SEQUENCE of three required fields.

```
Certificate ::= SEQUENCE {
    tbsCertificate          TBSCertificate,
    signatureAlgorithm      AlgorithmIdentifier,
    signatureValue          BIT STRING
}
```

- tbsCertificate

The field contains the names of the subject and issuer, a public key associated with the subject, a validity period, and other associated information. The tbsCertificate usually includes extensions which are described later.

```
TBSCertificate ::= SEQUENCE {
    version                [0] EXPLICIT Version DEFAULT v1,
    serialNumber           CertificateSerialNumber,
    signature              AlgorithmIdentifier,
    issuer                 Name (DN),
    validity               Validity,
    subject                Name (DN),
    subjectPublicKeyInfo   SubjectPublicKeyInfo,
    issuerUniqueID         [1] IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
    subjectUniqueID       [2] IMPLICIT UniqueIdentifier OPTIONAL,
                        -- If present, version MUST be v2 or v3
    extensions             [3] EXPLICIT Extensions OPTIONAL
                        -- If present, version MUST be v3
}
```

```
Version ::= INTEGER
```

```
CertificateSerialNumber ::= INTEGER
```

```
Validity ::= SEQUENCE {
    notBefore Time,
    notAfter  Time }

```

```
SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm      AlgorithmIdentifier,
    subjectPublicKey BIT STRING }

```

```
Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
```

```
Extension ::= SEQUENCE {
    extnID      OBJECT IDENTIFIER,
    critical    BOOLEAN DEFAULT FALSE,
    extnValue   OCTET STRING }

```

- signatureAlgorithm

The signatureAlgorithm field contains the identifier for the cryptographic algorithm used by the CA to sign this certificate. RFC4055 and RFC4491 list supported signature algorithms, but other signature algorithms MAY also be supported. An algorithm identifier is defined by the following ASN.1 structure:

```
AlgorithmIdentifier ::= SEQUENCE {
    algorithm      OBJECT IDENTIFIER,
    parameters    ANY DEFINED BY algorithm OPTIONAL }

```

The algorithm identifier is used to identify a cryptographic algorithm. The OBJECT IDENTIFIER component identifies the algorithm (such as DSA with SHA-1). The contents of the optional parameters field will vary according to the algorithm identified. This field MUST contain the same algorithm identifier as the signature field in the sequence tbsCertificate.

- signatureValue

The signatureValue field contains a digital signature computed upon the ASN.1 DER encoded tbsCertificate. The ASN.1 DER encoded tbsCertificate is used as the input to the signature function. This signature value is encoded as a BIT STRING and included in the signature field. The details of this process are specified for each of algorithms listed in RFC4055 and RFC4491.

By generating this signature, a CA certifies the validity of the information in the tbsCertificate field. In particular, the CA certifies the binding between the public key material and the subject of the certificate.

- Certificate Extensions (standard extensions)

The extensions defined for X.509 v3 certificates provide methods for associating additional attributes with users or public keys and for managing a certification hierarchy. The X.509 v3 certificate format also allows communities to define private extensions to carry information unique to those communities. Each extension in a certificate is designated as either critical or non-critical. A certificate using system MUST reject the certificate if it encounters a critical extension it does not recognize; however, a non-critical extension MAY be ignored if it is not recognized. The following sections present recommended extensions used within Internet certificates and standard locations for information. Communities may elect to use additional extensions; however, caution ought to be exercised in adopting any critical extensions in certificates which might prevent use in a general context.

- Authority Key Identifier

The authority key identifier extension provides a means of identifying the public key corresponding to the private key used to sign a certificate. This extension is used where an issuer has multiple signing keys (either due to multiple concurrent key pairs or due to changeover). The identification MAY be based on either the key identifier (the subject key identifier in the issuer's certificate) or on the issuer name and serial number.

```
AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier          [0] KeyIdentifier          OPTIONAL,
    authorityCertIssuer    [1] GeneralNames          OPTIONAL,
    authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL }
```

```
KeyIdentifier ::= OCTET STRING
```

- Subject Key Identifier

The subject key identifier extension provides a means of identifying certificates that contain a particular public key. To facilitate certification path construction, this extension MUST appear in all conforming CA certificates, that is, all certificates including the basic constraints extension (see below) where the value of cA is TRUE. The value of the subject key identifier MUST be the value placed in the key identifier field of the Authority Key Identifier extension (see above) of certificates issued by the subject of this certificate.

This extension MUST NOT be marked critical.

- Key Usage

The key usage extension defines the purpose (e.g., encipherment, signature, certificate signing) of the key contained in the certificate. The usage restriction might be employed when a key that could be used for more than one operation is to be restricted. For example, when an RSA key should be used only to verify signatures on objects other than public key certificates and CRLs, the digitalSignature and/or nonRepudiation bits would be asserted. Likewise, when an RSA key should be used only for key management, the keyEncipherment bit would be asserted.

This extension MUST appear in certificates that contain public keys that are used to validate digital signatures on other public key certificates or CRLs. When this extension appears, it SHOULD be marked critical.

```
KeyUsage ::= BIT STRING {
    digitalSignature      (0),
    nonRepudiation       (1),
    keyEncipherment      (2),
    dataEncipherment     (3),
    keyAgreement         (4),
    keyCertSign          (5),
    cRLSign              (6),
    encipherOnly         (7),
    decipherOnly         (8)
}
```

Bits in the KeyUsage type are used as follows:

- The digitalSignature bit is asserted when the subject public key is used with a digital signature mechanism to support security services other than certificate signing (bit 5), or CRL signing (bit 6). Digital signature mechanisms are often used for entity authentication and data origin authentication with integrity.
- The nonRepudiation bit is asserted when the subject public key is used to verify digital signatures used to provide a non-repudiation service which protects against the signing entity falsely denying some action, excluding certificate or CRL signing. In the case of later conflict, a reliable third party may determine the authenticity of the signed data. Further distinctions between the digitalSignature and nonRepudiation bits may be provided in specific certificate policies.
- The keyEncipherment bit is asserted when the subject public key is used for key transport. For example, when an RSA key is to be used for key management, then this bit is set.
- The dataEncipherment bit is asserted when the subject public key is used for enciphering user data, other than cryptographic keys.
- The keyAgreement bit is asserted when the subject public key is used for key agreement. For example, when a Diffie-Hellman key is to be used for key management, then this bit is set.

- The keyCertSign bit is asserted when the subject public key is used for verifying a signature on public key certificates. If the keyCertSign bit is asserted, then the cA bit in the basic constraints extension (see below) MUST also be asserted.
- The cRLSign bit is asserted when the subject public key is used for verifying a signature on certificate revocation list (e.g., a CRL, delta CRL, or an ARL). This bit MUST be asserted in certificates that are used to verify signatures on CRLs.
- The meaning of the encipherOnly bit is undefined in the absence of the keyAgreement bit. When the encipherOnly bit is asserted and the keyAgreement bit is also set, the subject public key may be used only for enciphering data while performing key agreement.
- The meaning of the decipherOnly bit is undefined in the absence of the keyAgreement bit. When the decipherOnly bit is asserted and the keyAgreement bit is also set, the subject public key may be used only for deciphering data while performing key agreement.

- Certificate Policies

The certificate policies extension contains a sequence of one or more policy information terms, each of which consists of an object identifier (OID) and optional qualifiers. Optional qualifiers, which MAY be present, are not expected to change the definition of the policy.

- Policy Mappings

This extension is used in CA certificates. It lists one or more pairs of OIDs; each pair includes an issuerDomainPolicy and a subjectDomainPolicy. The pairing indicates the issuing CA considers its issuerDomainPolicy equivalent to the subject CA's subjectDomainPolicy.

The issuing CA's users might accept an issuerDomainPolicy for certain applications. The policy mapping defines the list of policies associated with the subject CA that may be accepted as comparable to the issuerDomainPolicy.

This extension MAY be supported by CAs and/or applications, and it MUST be non-critical.

- Certificate Extensions (standard extensions) (cont'd)

- Subject Alternative Name

The subject alternative names extension allows additional identities to be bound to the subject of the certificate. Defined options include an Internet electronic mail address, a DNS name, an IP address, and a uniform resource identifier (URI). Other options exist, including completely local definitions. Multiple name forms, and multiple instances of each name form, MAY be included. Whenever such identities are to be bound into a certificate, the subject alternative name (or issuer alternative name) extension MUST be used.

Because the subject alternative name is considered to be definitively bound to the public key, all parts of the subject alternative name MUST be verified by the CA.

```
SubjectAltName ::= GeneralNames
```

```
GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName
```

```
GeneralName ::= CHOICE {
    otherName                [0]    OtherName,
    rfc822Name                [1]    IA5String,
    dNSName                   [2]    IA5String,
    x400Address                [3]    ORAddress,
    directoryName              [4]    Name,
    ediPartyName               [5]    EDIPartyName,
    uniformResourceIdentifier  [6]    IA5String,
    iPAddress                  [7]    OCTET STRING,
    registeredID               [8]    OBJECT IDENTIFIER }
```

- Issuer Alternative Names

This extension is used to associate Internet style identities with the certificate issuer. Issuer alternative names MUST be encoded as in Subject Alternative Name.

Where present, this extension SHOULD NOT be marked critical.

- Subject Directory Attributes

The subject directory attributes extension is used to convey identification attributes (e.g., nationality) of the subject. The extension is defined as a sequence of one or more attributes. This extension MUST be non-critical.

- Basic Constraints

The basic constraints extension identifies whether the subject of the certificate is a CA and the maximum depth of valid certification paths that include this certificate.

```
BasicConstraints ::= SEQUENCE {
    CA                      BOOLEAN DEFAULT FALSE,
    pathLenConstraint       INTEGER (0..MAX) OPTIONAL }
```

- Name Constraints

The name constraints extension, which MUST be used only in a CA certificate, indicates a name space within which all subject names in subsequent certificates in a certification path MUST be located. Restrictions apply to the subject distinguished name and apply to 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.

- Policy Constraints

The policy constraints extension can be used in certificates issued to CAs. The policy constraints extension constrains path validation in two ways. It can be used to prohibit policy mapping or require that each certificate in a path contain an acceptable policy identifier.

- Extended Key Usage

This extension indicates one or more purposes for which the certified public key may be used, in addition to or in place of the basic purposes indicated in the key usage extension. In general, this extension will appear only in end entity certificates. This extension is defined as follows:

```
ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId
KeyPurposeId ::= OBJECT IDENTIFIER
id-kp OBJECT IDENTIFIER ::=
id-kp-serverAuth          OBJECT IDENTIFIER ::=
-- TLS WWW server authentication
-- Key usage bits that may be consistent: digitalSignature,
-- keyEncipherment or keyAgreement
id-kp-clientAuth          OBJECT IDENTIFIER ::=
-- TLS WWW client authentication
-- Key usage bits that may be consistent: digitalSignature
-- and/or keyAgreement
id-kp-codeSigning         OBJECT IDENTIFIER ::=
-- Signing of downloadable executable code
-- Key usage bits that may be consistent: digitalSignature
id-kp-emailProtection     OBJECT IDENTIFIER ::=
-- E-mail protection
-- Key usage bits that may be consistent: digitalSignature,
-- nonRepudiation, and/or (keyEncipherment or keyAgreement)
id-kp-timeStamping        OBJECT IDENTIFIER ::=
-- Binding the hash of an object to a time
-- Key usage bits that may be consistent: digitalSignature
-- and/or nonRepudiation
id-kp-OCSPSigning         OBJECT IDENTIFIER ::=
-- Signing OCSP responses
-- Key usage bits that may be consistent: digitalSignature
-- and/or nonRepudiation
```

- CRL Distribution Points

The CRL distribution points extension identifies how CRL information is obtained. The extension SHOULD be non-critical, but this profile RECOMMENDS support for this extension by CAs and applications.

```
CRLDistributionPoints ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint
DistributionPoint ::= SEQUENCE {
    distributionPoint      [0]    DistributionPointName OPTIONAL,
    reasons                [1]    ReasonFlags OPTIONAL,
    cRLIssuer              [2]    GeneralNames OPTIONAL }
DistributionPointName ::= CHOICE {
    fullName               [0]    GeneralNames,
    nameRelativeToCRLIssuer [1]    RelativeDistinguishedName }
ReasonFlags ::= BIT STRING {
    unused                 (0),
    keyCompromise         (1),
    cACompromise          (2),
    affiliationChanged    (3),
    superseded            (4),
    cessationOfOperation (5),
    certificateHold       (6),
```

```

        privilegeWithdrawn      (7),
        aACompromise           (8) }

```

## 2.2. CRL (Certificate Revocation List)

A complete CRL lists all unexpired certificates, within its scope, that have been revoked for one of the revocation reasons covered by the CRL scope. The CRL issuer MAY also generate delta CRLs. A delta CRL only lists those certificates, within its scope, whose revocation status has changed since the issuance of a referenced complete CRL. The referenced complete CRL is referred to as a base CRL. The scope of a delta CRL MUST be the same as the base CRL that it references.

```

CertificateList ::= SEQUENCE {
    tbsCertList      TBSCertList,
    signatureAlgorithm AlgorithmIdentifier,
    signatureValue   BIT STRING }

```

- tbsCertList

```

TBSCertList ::= SEQUENCE {
    version          Version OPTIONAL,
                    -- if present, MUST be v2
    signature        AlgorithmIdentifier,
    issuer           Name,
    thisUpdate       Time,
    nextUpdate       Time OPTIONAL,
    revokedCertificates SEQUENCE OF SEQUENCE {
        userCertificate CertificateSerialNumber,
        revocationDate  Time,
        crlEntryExtensions Extensions OPTIONAL
                    -- if present, MUST be v2
    } OPTIONAL,
    crlExtensions    [0] EXPLICIT Extensions OPTIONAL
                    -- if present, MUST be v2
}

```

- CRL Entry Extensions
  - Reason Code

```

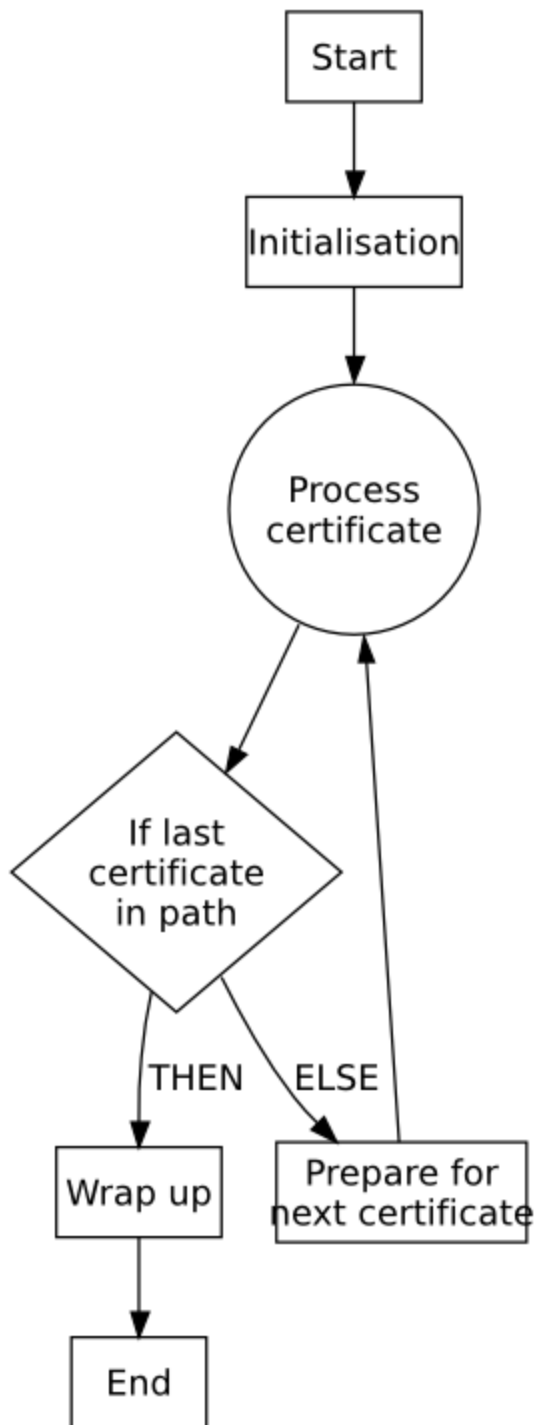
reasonCode ::=

CRLReason ::= ENUMERATED {
    unspecified          (0),
    keyCompromise       (1),
    cACompromise        (2),
    affiliationChanged   (3),
    superseded          (4),
    cessationOfOperation (5),
    certificateHold      (6),
    removeFromCRL       (8),
    privilegeWithdrawn  (9),
    aACompromise        (10) }

```

## 2.3. Certification Path Validation

Certification path validation procedures for the Internet PKI are based on the algorithm supplied in [X.509]. Certification path processing verifies the binding between the subject distinguished name and/or subject alternative name and subject public key. The binding is limited by constraints which are specified in the certificates which comprise the path and inputs which are specified by the relying party. The basic constraints and policy constraints extensions allow the certification path processing logic to automate the decision making process.



- Basic Certificate Processing

The basic path processing actions to be performed for certificate  $i$  (for all  $i$  in  $[1..n]$ ).

- verify the basic certificate information. The certificate MUST satisfy each of the following:
  - The certificate was signed with the `working_public_key_algorithm` using the `working_public_key` and the `working_public_key_parameters`.
  - The certificate validity period includes the current time.
  - At the current time, the certificate is not revoked and is not on hold status. This may be determined by obtaining the appropriate CRL status information, or by out-of-band mechanisms.
  - The certificate issuer name is the `working_issuer_name`.

### 3. Online Certificate Status Protocol (OCSP) - RFC 2560

In lieu of or as a supplement to checking against a periodic CRL, it may be necessary to obtain timely information regarding the revocation status of a certificate. Examples include high-value funds transfer or large stock trades.

The Online Certificate Status Protocol (OCSP) enables applications to determine the (revocation) state of an identified certificate. OCSP may be used to satisfy some of the operational requirements of providing more timely revocation information than is possible with CRLs and may also be used to obtain additional status information. An OCSP client issues a status request to an OCSP responder and suspends acceptance of the certificate in question until the responder provides a response.

This protocol specifies the data that needs to be exchanged between an application checking the status of a certificate and the server providing that status.

- An OCSP request contains the following data:
  - protocol version
  - service request
  - target certificate identifier
  - optional extensions which MAY be processed by the OCSP Responder
- Upon receipt of a request, an OCSP Responder determines if:
  1. the message is well formed
  2. the responder is configured to provide the requested service and
  3. the request contains the information needed by the responder

If any one of the prior conditions are not met, the OCSP responder produces an error message; otherwise, it returns a definitive response.

OCSP responses can be of various types. An OCSP response consists of a response type and the bytes of the actual response. There is one basic type of OCSP response that MUST be supported by all OCSP servers and clients. The rest of this section pertains only to this basic response type.

- All definitive response messages SHALL be digitally signed. The key used to sign the response MUST belong to one of the following:
  - the CA who issued the certificate in question
  - a Trusted Responder whose public key is trusted by the requester
  - a CA Designated Responder (Authorized Responder) who holds a
  - specially marked certificate issued directly by the CA, indicating that the responder may issue OCSP responses for that CA
- A definitive response message is composed of:
  - version of the response syntax
  - name of the responder
  - responses for each of the certificates in a request
  - optional extensions
  - signature algorithm OID
  - signature computed across hash of the response
- The response for each of the certificates in a request consists of
  - target certificate identifier
  - certificate status value
  - response validity interval
  - optional extensions
- This specification defines the following definitive response indicators for use in the certificate status value:
  - good
 

The "good" state indicates a positive response to the status inquiry. At a minimum, this positive response indicates that the certificate is not revoked, but does not necessarily mean that the certificate was ever issued or that the time at which the response was produced is within the certificate's validity interval. Response extensions may be used to convey additional information on assertions made by the responder regarding the status of the certificate such as positive statement about issuance, validity, etc.
  - revoked
 

The "revoked" state indicates that the certificate has been revoked (either permanently or temporarily (on hold)).
  - unknown
 

The "unknown" state indicates that the responder doesn't know about the certificate being requested.
- Functional Requirements :
  - Certificate Content
 

In order to convey to OCSP clients a well-known point of information access, CAs SHALL provide the capability to include the AuthorityInfoAccess extension in certificates that can be checked using OCSP. Alternatively, the accessLocation for the OCSP provider may be configured locally at the OCSP client.

CAs that support an OCSP service, either hosted locally or provided by an Authorized Responder, MUST

provide for the inclusion of a value for a uniformResourceIndicator (URI) accessLocation and the OID value id-ad-ocsp for the accessMethod in the AccessDescription SEQUENCE.

The value of the accessLocation field in the subject certificate defines the transport (e.g. HTTP) used to access the OCSP responder and may contain other transport dependent information (e.g. a URL).

- Signed Response Acceptance Requirements

Prior to accepting a signed response as valid, OCSP clients SHALL confirm that:

1. The certificate identified in a received response corresponds to that which was identified in the corresponding request;
2. The signature on the response is valid;
3. The identity of the signer matches the intended recipient of the request.
4. The signer is currently authorized to sign the response.
5. The time at which the status being indicated is known to be correct (thisUpdate) is sufficiently recent.
6. When available, the time at or before which newer information will be available about the status of the certificate (nextUpdate) is greater than the current time.

### 3.1. Request Syntax

```

OCSPRequest ::= SEQUENCE {
    tbsRequest          TBSRequest,
    optionalSignature [0] EXPLICIT Signature OPTIONAL }

TBSRequest ::= SEQUENCE {
    version          [0] EXPLICIT Version DEFAULT v1,
    requestorName   [1] EXPLICIT GeneralName OPTIONAL,
    requestList     SEQUENCE OF Request,
    requestExtensions [2] EXPLICIT Extensions OPTIONAL }

Signature ::= SEQUENCE {
    signatureAlgorithm AlgorithmIdentifier,
    signature          BIT STRING,
    certs              [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL }

Version ::= INTEGER

Request ::= SEQUENCE {
    reqCert          CertID,
    singleRequestExtensions [0] EXPLICIT Extensions OPTIONAL }

CertID ::= SEQUENCE {
    hashAlgorithm      AlgorithmIdentifier,
    issuerNameHash     OCTET STRING, -- Hash of Issuer's DN
    issuerKeyHash      OCTET STRING, -- Hash of Issuers public key
    serialNumber       CertificateSerialNumber }

```

### 3.2. OCSP Response

An OCSP response at a minimum consists of a responseStatus field indicating the processing status of the prior request. If the value of responseStatus is one of the error conditions, responseBytes are not set.

```

OCSPResponse ::= SEQUENCE {
    responseStatus  OCSPResponseStatus,
    responseBytes   [0] EXPLICIT ResponseBytes OPTIONAL }

OCSPResponseStatus ::= ENUMERATED {
    successful          (0), --Response has valid confirmations
    malformedRequest   (1), --Illegal confirmation request
    internalError      (2), --Internal error in issuer
    tryLater           (3), --Try again later
                       --(4) is not used
    sigRequired        (5), --Must sign the request
    unauthorized       (6)  --Request unauthorized }

```

## 4. Certificate Policy and Certification Practices Framework - RFC 3647

### 4.1. Certificate policy (CP)

The X.509 standard defines a CP as "a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements". An X.509 Version 3

certificate may identify a specific applicable CP, which may be used by a relying party to decide whether or not to trust a certificate, associated public key, or any digital signatures verified using the public key for a particular purpose.

When a certification authority issues a certificate, it is providing a statement to a certificate user (i.e., a relying party) that a particular public key is bound to the identity and/or other attributes of a particular entity (the certificate subject, which is usually also the subscriber). The extent to which the relying party should rely on that statement by the CA, however, needs to be assessed by the relying party or entity controlling or coordinating the way relying parties or relying party applications use certificates. Different certificates are issued following different practices and procedures, and may be suitable for different applications and/or purposes.

CPs typically fall into two major categories. First, some CPs "indicate the applicability of a certificate to a particular community". These CPs set forth requirements for certificate usage and requirements on members of a community. For instance, a CP may focus on the needs of a geographical community, such as the ETSI policy requirements for CAs issuing qualified certificates.

Also, a CP of this kind may focus on the needs of a specific vertical-market community, such as financial services. The second category of typical CPs "indicate the applicability of a certificate to a ... class of application with common security requirements." These CPs identify a set of applications or uses for certificates and say that these applications or uses require a certain level of security. They then set forth PKI requirements that are appropriate for these applications or uses. A CP within this category often sets requirements appropriate for a certain "level of assurance" provided by certificates, relative to certificates issued pursuant to related CPs. These levels of assurance may correspond to "classes" or "types" of certificates.

A CP is represented in a certificate by a unique number called an "Object Identifier" (OID).

CPs also constitute a basis for an audit, accreditation, or another assessment of a CA. Each CA can be assessed against one or more certificate policies or CPSs that it is recognized as implementing. When one CA issues a CA-certificate for another CA, the issuing CA must assess the set of certificate policies for which it trusts the subject CA (such assessment may be based upon an assessment with respect to the certificate policies involved). The assessed set of certificate policies is then indicated by the issuing CA in the CA-certificate. The X.509 certification path processing logic employs these CP indications in its well-defined trust model.

## 4.2. Certification Practice Statement (CPS)

The term certification practice statement (CPS) is defined as: "A statement of the practices which a certification authority employs in issuing certificates". A CPS establishes practices concerning lifecycle services in addition to issuance, such as certificate management (including publication and archiving), revocation, and renewal or re-keying.

*"A certification practice statement may take the form of a declaration by the certification authority of the details of its trustworthy system and the practices it employs in its operations and in support of issuance of a certificate ..."*

This form of CPS is the most common type, and can vary in length and level of detail.

Some PKIs may not have the need to create a thorough and detailed statement of practices. For example, the CA may itself be the relying party and would already be aware of the nature and trustworthiness of its services. In other cases, a PKI may provide certificates providing only a very low level of assurances where the applications being secured may pose only marginal risks if compromised. In these cases, an organization establishing a PKI may only want to write or have CAs use a subscriber agreement, relying party agreement, or agreement combining subscriber and relying party terms, depending on the role of the different PKI participants. In such a PKI, that agreement may serve as the only "statement of practices" used by one or more CAs within that PKI. Consequently, that agreement may also be considered a CPS and can be entitled or subtitled as such.

Likewise, since a detailed CPS may contain sensitive details of its system, a CA may elect not to publish its entire CPS. It may instead opt to publish a CPS Summary (or CPS Abstract). The CPS Summary would contain only those provisions from the CPS that the CA considers to be relevant to the participants in the PKI (such as the responsibilities of the parties or the stages of the certificate lifecycle). A CPS Summary, however, would not contain those sensitive provisions of the full CPS that might provide an attacker with useful information about the CA's operations. Throughout this document, the use of "CPS" includes both a detailed CPS and a CPS Summary (unless otherwise specified).

CPSs do not automatically constitute contracts and do not automatically bind PKI participants as a contract would. Where a document serves the dual purpose of being a subscriber or relying party agreement and CPS, the document is intended to be a contract and constitutes a binding contract to the extent that a subscriber or relying party agreement would ordinarily be considered as such. Most CPSs, however, do not serve such a dual purpose. Therefore, in most cases, a CPS's terms have a binding effect as contract terms only if a separate document creates a contractual relationship between the parties and that document incorporates part or all of the CPS by reference. Further, if a particular PKI employs a CPS Summary (as opposed to the entire CPS), the CPS Summary could be incorporated into any applicable subscriber or relying party agreement.

In the future, a court or applicable statutory or regulatory law may declare that a certificate itself is a document that is capable of creating a contractual relationship, to the extent its mechanisms designed for incorporation by reference (such as the Certificate Policies extension and its qualifiers) indicate that terms of its use appear in certain documents. In the meantime, however, some subscriber agreements and relying party agreements may incorporate a CPS by reference and therefore make its terms binding on the parties to such agreements.

- The main differences between CPs and CPSs can therefore be summarized as follows:

1. A PKI uses a CP to establish requirements that state what participants within it must do. A single CA or organization can use a CPS to disclose how it meets the requirements of a CP or how it implements its practices and controls.
2. A CP facilitates interoperation through cross-certification, unilateral certification, or other means. Therefore, it is intended to cover multiple CAs. By contrast, a CPS is a statement of a single CA or organization. Its purpose is not to facilitate interoperation (since doing so is the function of a CP).
3. A CPS is generally more detailed than a CP and specifies how the CA meets the requirements specified in the one or more CPs under which it issues certificates.

In addition to populating the certificate policies extension with the applicable CP object identifier, a certification authority may include, in certificates it issues, a reference to its certification practice statement.

### 4.3. Recommended CP or CPS outline

In order to comply with the RFC, the drafters of a compliant CP or CPS are strongly advised to adhere to the following outline:

#### 1. INTRODUCTION

1. Overview
2. Document name and identification
3. PKI participants
  1. Certification authorities
  2. Registration authorities
  3. Subscribers
  4. Relying parties
  5. Other participants
4. Certificate usage
  1. Appropriate certificate uses
  2. Prohibited certificate uses
5. Policy administration
  1. Organization administering the document
  2. Contact person
  3. Person determining CPS suitability for the policy
  4. CPS approval procedures
6. Definitions and acronyms

#### 2. PUBLICATION AND REPOSITORY RESPONSIBILITIES

1. Repositories
2. Publication of certification information
3. Time or frequency of publication
4. Access controls on repositories

#### 3. IDENTIFICATION AND AUTHENTICATION

1. Naming
  1. Types of names
  2. Need for names to be meaningful
  3. Anonymity or pseudonymity of subscribers
  4. Rules for interpreting various name forms
  5. Uniqueness of names
  6. Recognition, authentication, and role of trademarks
2. Initial identity validation
  1. Method to prove possession of private key
  2. Authentication of organization identity
  3. Authentication of individual identity
  4. Non-verified subscriber information
  5. Validation of authority
  6. Criteria for interoperation
3. Identification and authentication for re-key requests
  1. Identification and authentication for routine re-key
  2. Identification and authentication for re-key after revocation
4. Identification and authentication for revocation request

#### 4. CERTIFICATE LIFE-CYCLE OPERATIONAL REQUIREMENTS

1. Certificate Application
  1. Who can submit a certificate application
  2. Enrollment process and responsibilities
2. Certificate application processing
  1. Performing identification and authentication functions
  2. Approval or rejection of certificate applications
  3. Time to process certificate applications
3. Certificate issuance
  1. CA actions during certificate issuance
  2. Notification to subscriber by the CA of issuance of certificate
4. Certificate acceptance
  1. Conduct constituting certificate acceptance

2. Publication of the certificate by the CA
3. Notification of certificate issuance by the CA to other entities
5. Key pair and certificate usage
  1. Subscriber private key and certificate usage
  2. Relying party public key and certificate usage
6. Certificate renewal
  1. Circumstance for certificate renewal
  2. Who may request renewal
  3. Processing certificate renewal requests
  4. Notification of new certificate issuance to subscriber
  5. Conduct constituting acceptance of a renewal certificate
  6. Publication of the renewal certificate by the CA
  7. Notification of certificate issuance by the CA to other entities
7. Certificate re-key
  1. Circumstance for certificate re-key
  2. Who may request certification of a new public key
  3. Processing certificate re-keying requests
  4. Notification of new certificate issuance to subscriber
  5. Conduct constituting acceptance of a re-keyed certificate
  6. Publication of the re-keyed certificate by the CA
  7. Notification of certificate issuance by the CA to other entities
8. Certificate modification
  1. Circumstance for certificate modification
  2. Who may request certificate modification
  3. Processing certificate modification requests
  4. Notification of new certificate issuance to subscriber
  5. Conduct constituting acceptance of modified certificate
  6. Publication of the modified certificate by the CA
  7. Notification of certificate issuance by the CA to other entities
9. Certificate revocation and suspension
  1. Circumstances for revocation
  2. Who can request revocation
  3. Procedure for revocation request
  4. Revocation request grace period
  5. Time within which CA must process the revocation request
  6. Revocation checking requirement for relying parties
  7. CRL issuance frequency (if applicable)
  8. Maximum latency for CRLs (if applicable)
  9. On-line revocation/status checking availability
  10. On-line revocation checking requirements
  11. Other forms of revocation advertisements available
  12. Special requirements re key compromise
  13. Circumstances for suspension
  14. Who can request suspension
  15. Procedure for suspension request
  16. Limits on suspension period
10. Certificate status services
  1. Operational characteristics
  2. Service availability
  3. Optional features
  4. End of subscription
  5. Key escrow and recovery
  6. Key escrow and recovery policy and practices
  7. Session key encapsulation and recovery policy and practices

## 5. FACILITY, MANAGEMENT, AND OPERATIONAL CONTROLS

1. Physical controls
  1. Site location and construction
  2. Physical access
  3. Power and air conditioning
  4. Water exposures
  5. Fire prevention and protection
  6. Media storage
  7. Waste disposal
  8. Off-site backup
2. Procedural controls
  1. Trusted roles
  2. Number of persons required per task
  3. Identification and authentication for each role
  4. Roles requiring separation of duties
3. Personnel controls
  1. Qualifications, experience, and clearance requirements
  2. Background check procedures
  3. Training requirements
  4. Retraining frequency and requirements
  5. Job rotation frequency and sequence
  6. Sanctions for unauthorized actions
  7. Independent contractor requirements
  8. Documentation supplied to personnel

4. Audit logging procedures
  1. Types of events recorded
  2. Frequency of processing log
  3. Retention period for audit log
  4. Protection of audit log
  5. Audit log backup procedures
  6. Audit collection system (internal vs. external)
  7. Notification to event-causing subject
  8. Vulnerability assessments
5. Records archival
  1. Types of records archived
  2. Retention period for archive
  3. Protection of archive
  4. Archive backup procedures
  5. Requirements for time-stamping of records
  6. Archive collection system (internal or external)
  7. Procedures to obtain and verify archive information
6. Key changeover
7. Compromise and disaster recovery
  1. Incident and compromise handling procedures
  2. Computing resources, software, and/or data are corrupted
  3. Entity private key compromise procedures
  4. Business continuity capabilities after a disaster
8. CA or RA termination

## 6. TECHNICAL SECURITY CONTROLS

1. Key pair generation and installation
  1. Key pair generation
  2. Private key delivery to subscriber
  3. Public key delivery to certificate issuer
  4. CA public key delivery to relying parties
  5. Key sizes
  6. Public key parameters generation and quality checking
  7. Key usage purposes (as per X.509 v3 key usage field)
2. Private Key Protection and Cryptographic Module Engineering Controls
  1. Cryptographic module standards and controls
  2. Private key (n out of m) multi-person control
  3. Private key escrow
  4. Private key backup
  5. Private key archival
  6. Private key transfer into or from a cryptographic module
  7. Private key storage on cryptographic module
  8. Method of activating private key
  9. Method of deactivating private key
  10. Method of destroying private key
  11. Cryptographic Module Rating
3. Other aspects of key pair management
  1. Public key archival
  2. Certificate operational periods and key pair usage periods
4. Activation data
  1. Activation data generation and installation
  2. Activation data protection
  3. Other aspects of activation data
5. Computer security controls
  1. Specific computer security technical requirements
  2. Computer security rating
6. Life cycle technical controls
  1. System development controls
  2. Security management controls
  3. Life cycle security controls
7. Network security controls
8. Time-stamping

## 7. CERTIFICATE, CRL, AND OCSP PROFILES

1. Certificate profile
  1. Version number(s)
  2. Certificate extensions
  3. Algorithm object identifiers
  4. Name forms
  5. Name constraints
  6. Certificate policy object identifier
  7. Usage of Policy Constraints extension
  8. Policy qualifiers syntax and semantics
  9. Processing semantics for the critical Certificate Policies extension
2. CRL profile
  1. Version number(s)
  2. CRL and CRL entry extensions
3. OCSP profile

1. Version number(s)
2. OCSP extensions

## 8. COMPLIANCE AUDIT AND OTHER ASSESSMENTS

1. Frequency or circumstances of assessment
2. Identity/qualifications of assessor
3. Assessor's relationship to assessed entity
4. Topics covered by assessment
5. Actions taken as a result of deficiency
6. Communication of results

## 9. OTHER BUSINESS AND LEGAL MATTERS

1. Fees
  1. Certificate issuance or renewal fees
  2. Certificate access fees
  3. Revocation or status information access fees
  4. Fees for other services
  5. Refund policy
2. Financial responsibility
  1. Insurance coverage
  2. Other assets
  3. Insurance or warranty coverage for end-entities
3. Confidentiality of business information
  1. Scope of confidential information
  2. Information not within the scope of confidential information
  3. Responsibility to protect confidential information
4. Privacy of personal information
  1. Privacy plan
  2. Information treated as private
  3. Information not deemed private
  4. Responsibility to protect private information
  5. Notice and consent to use private information
  6. Disclosure pursuant to judicial or administrative process
  7. Other information disclosure circumstances
5. Intellectual property rights
6. Representations and warranties
  1. CA representations and warranties
  2. RA representations and warranties
  3. Subscriber representations and warranties
  4. Relying party representations and warranties
  5. Representations and warranties of other participants
7. Disclaimers of warranties
8. Limitations of liability
9. Indemnities
10. Term and termination
  1. Term
  2. Termination
  3. Effect of termination and survival
11. Individual notices and communications with participants
12. Amendments
  1. Procedure for amendment
  2. Notification mechanism and period
  3. Circumstances under which OID must be changed
13. Dispute resolution provisions
14. Governing law
15. Compliance with applicable law
16. Miscellaneous provisions
  1. Entire agreement
  2. Assignment
  3. Severability
  4. Enforcement (attorneys' fees and waiver of rights)
  5. Force Majeure
17. Other provisions

## 5. Time-Stamping Authorities (TSAs) - RFC 3628

In creating reliable and manageable digital evidence it is necessary to have an agreed upon method of associating time data to transaction so that they might be compared to each other at a later time. The quality of this evidence is based on creating and managing the data structure that represent the events and the quality of the parametric data points that anchor them to the real world. In this instance this being the time data and how it was applied.

A typical transaction is a digitally signed document, where it is necessary to prove that the digital signature from the signer was applied when the signer's certificate was valid.

A timestamp or a time mark (which is an audit record kept in a secure audit trail from a trusted third party) 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) was applied before the end of the validity period of the signer's certificate,
2. the time-stamp (or time mark) was 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 proves the validity of a digital signature over the whole of any certificate chain.

The electronic time stamp is gaining interest from the business sector as an important component of electronic signatures. It is also featured by the ETSI Electronic Signature Format standard (TS 101 733) or Electronic Signature Formats for long term electronic signatures (RFC 3126), built upon the Time-Stamp Protocol (RFC 3161). Agreed minimum security and quality requirements are necessary in order to ensure trustworthy validation of long-term electronic signatures.

The European Directive 1999/93/EC defines certification service provider as "an entity or a legal or natural person who issues certificates or provides other services related to electronic signatures". One example of a certification-service-provider is a Time-Stamping Authority.

## 5.1. Time-Stamp Token

The TSA shall ensure that time-stamp tokens are issued securely and include the correct time. In particular:

1. The time-stamp token shall include an identifier for the time-stamp policy;
2. Each time-stamp token shall have a unique identifier;
3. The time values the TSU uses in the time-stamp token shall be traceable to at least one of the real time values distributed by a UTC(k) laboratory.

NOTE: The Bureau International des Poids et Mesures (BIPM) computes UTC on the basis of its local representations UTC(k) from a large ensemble of atomic clocks in national metrology institutes and national astronomical observatories round the world. The BIPM disseminates UTC through its monthly Circular T. This is available on the BIPM website ([www.bipm.org](http://www.bipm.org)) and it officially identifies all those institutes having recognized UTC(k) time scales.

4. The time included in the time-stamp token shall be synchronized with UTC within the accuracy defined in this policy and, if present, within the accuracy defined in the time-stamp token itself;
5. If the time-stamp provider's clock is detected as being out of the stated accuracy then time-stamp tokens shall not be issued.
6. The time-stamp token shall include a representation (e.g., hash value) of the datum being time-stamped as provided by the requestor;
7. The time-stamp token shall be signed using a key generated exclusively for this purpose.

NOTE: A protocol for a time-stamp token is defined in RFC 3631 and profiled in TS 101 861.

NOTE: In the case of a number of requests at approximately the same time, the ordering of the time within the accuracy of the TSU clock is not mandated.

8. The time-stamp token shall include:
  - where applicable, an identifier for the country in which the TSA is established;
  - an identifier for the TSA;
  - an identifier for the unit which issues the time-stamps.

## 5.2. Clock Synchronization with UTC

The TSA shall ensure that its clock is synchronized with UTC within the declared accuracy. In particular:

1. The calibration of the TSU clocks shall be maintained such that the clocks shall not be expected to drift outside the declared accuracy.
2. The TSU clocks shall be protected against threats which could result in an undetected change to the clock that takes it outside its calibration.

NOTE: Threats may include tampering by unauthorized personnel, radio or electrical shocks.

3. The TSA shall ensure that, if the time that would be indicated in a time-stamp token drifts or jumps out of synchronization with UTC, this will be detected.

NOTE: Relying parties are required to be informed of such events.

4. The TSA shall ensure that clock synchronization is maintained when a leap second occurs as notified by the appropriate body. The change to take account of the leap second shall occur during the last minute of the day when the leap second is scheduled to occur. A record shall be maintained of the exact time (within the declared accuracy) when this change occurred.

NOTE: A leap second is an adjustment to UTC by skipping or adding an extra second on the last second of a UTC month. First preference is given to the end of December and June, and second preference is given to the end of March and September.

### 5.3. Time-Stamp Protocol (TSP) - RFC 3161

A time-stamping service supports assertions of proof that a datum existed before a particular time. A TSA may be operated as a Trusted Third Party (TTP) service, though other operational models may be appropriate, e.g., an organization might require a TSA for internal time-stamping purposes.

Non-repudiation services require the ability to establish the existence of data before specified times. This protocol may be used as a building block to support such services.

- The TSA is REQUIRED:
  1. to use a trustworthy source of time.
  2. to include a trustworthy time value for each time-stamp token.
  3. to include a unique integer for each newly generated time-stamp token.
  4. to produce a time-stamp token upon receiving a valid request from the requester, when it is possible.
  5. to include within each time-stamp token an identifier to uniquely indicate the security policy under which the token was created.
  6. to only time-stamp a hash representation of the datum, i.e., a data imprint associated with a one-way collision resistant hash-function uniquely identified by an OID.
  7. to examine the OID of the one-way collision resistant hash-function and to verify that the hash value length is consistent with the hash algorithm.
  8. not to examine the imprint being time-stamped in any way (other than to check its length, as specified in the previous bullet).
  9. not to include any identification of the requesting entity in the time-stamp tokens.
  10. to sign each time-stamp token using a key generated exclusively for this purpose and have this property of the key indicated on the corresponding certificate.
  11. to include additional information in the time-stamp token, if asked by the requester using the extensions field, only for the extensions that are supported by the TSA. If this is not possible, the TSA SHALL respond with an error message.

### 5.4. Request Format

```

TimeStampReq ::= SEQUENCE {
    version                INTEGER ,
    messageImprint        MessageImprint,
    --a hash algorithm OID and the hash value of the data to be
    --time-stamped
    reqPolicy              TSAPolicyId          OPTIONAL,
    nonce                  INTEGER              OPTIONAL,
    certReq                BOOLEAN              DEFAULT FALSE,
    extensions              [0] IMPLICIT Extensions OPTIONAL }

MessageImprint ::= SEQUENCE {
    hashAlgorithm          AlgorithmIdentifier,
    hashedMessage          OCTET STRING }

```

The messageImprint field SHOULD contain the hash of the datum to be time-stamped. The hash is represented as an OCTET STRING. Its length MUST match the length of the hash value for that algorithm (e.g., 20 bytes for SHA-1 or 16 bytes for MD5).

The reqPolicy field, if included, indicates the TSA policy under which the TimeStampToken SHOULD be provided.

The nonce, if included, allows the client to verify the timeliness of the response when no local clock is available. The nonce is a large random number with a high probability that the client generates it only once (e.g., a 64 bit integer). In such a case the same nonce value MUST be included in the response, otherwise the response shall be rejected.

If the certReq field is present and set to true, the TSA's public key certificate that is referenced by the ESSCertID identifier inside a SigningCertificate attribute in the response MUST be provided by the TSA in the certificates field from the SignedData structure in that response. That field may also contain other certificates.

If the certReq field is missing or if the certReq field is present and set to false then the certificates field from the SignedData structure MUST not be present in the response.

The time-stamp request does not identify the requester, as this information is not validated by the TSA. In situations where the TSA requires the identity of the requesting entity, alternate identification/authentication means have to be used.

### 5.5. Response Format

```

TimeStampResp ::= SEQUENCE {
    status                  PKIStatusInfo,
    timeStampToken          TimeStampToken  OPTIONAL }

PKIStatusInfo ::= SEQUENCE {

```

```

status          PKIStatus,
statusString    PKIFreeText    OPTIONAL,
failInfo        PKIFailureInfo OPTIONAL }

```

When the status contains the value zero or one, a TimeStampToken MUST be present. When status contains a value other than zero or one, a TimeStampToken MUST NOT be present. One of the following values MUST be contained in status:

```

PKIStatus ::= INTEGER {
  granted          (0),
  -- when the PKIStatus contains the value zero a TimeStampToken, as
  requested, is present.
  grantedWithMods (1),
  -- when the PKIStatus contains the value one a TimeStampToken,
  with modifications, is present.
  rejection       (2),
  waiting         (3),
  revocationWarning (4),
  -- this message contains a warning that a revocation is
  -- imminent
  revocationNotification (5)
  -- notification that a revocation has occurred }

```

## 6. Public-Key Cryptography Standards (PKCS)

	Version	Name	Comments
PKCS#1	2.1	RSA Cryptography Standard	See RFC 3447. Defines the format of RSA encryption.
PKCS#3	1.4	Diffie-Hellman Key Agreement Standard	A cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel.
PKCS#5	2.0	Password-based Encryption Standard	See RFC 2898 and PBKDF2.
PKCS#6	1.5	Extended-Certificate Syntax Standard	Defines extensions to the old v1 X.509 certificate specification. Obsoleted by v3 of the same.
PKCS#7	1.5	Cryptographic Message Syntax Standard	See RFC 2315. Used to sign and/or encrypt messages under a PKI. Used also for certificate dissemination (for instance as a response to a PKCS#10 message). Formed the basis for S/MIME, which is now based on RFC 3852, an updated Cryptographic Message Syntax Standard (CMS).
PKCS#8	1.2	Private-Key Information Syntax Standard	
PKCS#9	2.0	Selected Attribute Types	Defines selected attribute types for use in PKCS #6 extended certificates, PKCS #7 digitally signed messages, PKCS #8 private-key information, and PKCS #10 certificate-signing requests.
<b>PKCS#10</b>	<b>1.7</b>	<b>Certification Request Standard</b>	<b>See RFC 2986. Format of messages sent to a certification authority to request certification of a public key. See certificate signing request.</b>
PKCS#11	2.20	Cryptographic Token Interface (Cryptoki)	An API defining a generic interface to cryptographic tokens (see also Hardware Security Module).
PKCS#12	1.0	Personal Information Exchange Syntax Standard	Defines a file format commonly used to store private keys with accompanying public key certificates, protected with a password-based symmetric key.
PKCS#13	–	Elliptic Curve Cryptography Standard	(Under development)
PKCS#14	–	Pseudo-random Number Generation	(Under development)
PKCS#15	1.1	Cryptographic Token Information Format Standard	Defines a standard allowing users of cryptographic tokens to identify themselves to applications, independent of the application's Cryptoki implementation (PKCS #11) or other API. RSA has relinquished IC-card-related parts of this standard to ISO/IEC 7816-15.[1]

## 7. Bibliographic references

- [PKCS \(wikipedia\)](#)
- [EESSI](#)
- [CEN CWA on electronic signatures](#)