INTERNATIONAL STANDARD

ISO 15782-2

First edition
2001-11-01

Banking — Certificate management —

Part 2:
Certificate extensions

Banque — Gestion des certificats —

Partie 2: Extensions des certificats

Reference number
ISO 15782-2:2001(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO 15782 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

International Standard ISO 15782-2 was prepared by Technical Committee ISO/TC 68, *Banking, securities and other financial services*, Subcommittee SC 2, *Security management and general banking operations*.

ISO 15782 consists of the following parts, under the general title *Banking — Certificate management*:

— *Part 1: Public key certificates*

— *Part 2: Certificate extensions*

Annex A of this part of ISO 15782 is for information only.

Introduction

This part of ISO 15782 extracts and adopts selected definitions of certificate extensions from ISO 9594-8 and adds control requirements and other information required for financial institution use.

While the techniques specified in this part of ISO 15782 are designed to maintain the integrity of financial messages, the Standard does not guarantee that a particular implementation is secure. It is the responsibility of the financial institution to put an overall process in place with the necessary controls to ensure that the process is securely implemented. Furthermore, the controls should include the application of appropriate audit tests in order to validate compliance.

The binding association between the identity of the owner of a public key and that key shall be documented in order to prove the ownership of a public key. This binding is called a “public key certificate”. Public key certificates are generated by a trusted third entity known as a Certification Authority (CA).

Banking — Certificate management —

Part 2: **Certificate extensions**

1 Scope

This part of ISO 15782

— extracts and adopts selected definitions of certificate extensions from ISO/IEC 9594-8;

— specifies additional requirements when certificate extensions are used by the financial services industry.

This part of ISO 15782 is to be used with financial institution standards, including ISO 15782-1.

NOTE Distinguished Encoding Rules (DER) of ASN.1 for encoding of ASN.1-defined certificate extensions are specified in ISO/IEC 8825-1. The DER rules defined by ISO/IEC 9594-8 are incomplete and can lead to ambiguities when encoding some values.

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO 15782. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO 15782 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

ISO/IEC 9594-2 | ITU-T Recommendation X.501, *Information technology — Open Systems Interconnection — The Directory: Models*

ISO/IEC 9594-8:1998 | ITU-T Recommendation X.509 (1997), *Information technology — Open Systems Interconnection — The Directory: Authentication framework*

ISO/IEC 9834-1 | CCITT Recommendation X.660 *Information technology — Open Systems Interconnection — Procedures for the operation of OSI Registration Authorities: General procedures*

ISO/IEC 10021-4 | ITU-T Recommendation X.411, *Information technology — Message Handling Systems (MHS) — Message transfer system: Abstract service definition and procedures*

ISO 15782-1:—[1], *Banking — Certificate management — Part 1: Public key certificates*

RFC 791:1981[2], *Internet protocol*

1) To be published.

2) Obsoletes RFC 760; obsoleted by RFC 1060.

RFC 822:1982[3], *Standard for the format of ARPA Internet text messages*

RFC 1035:1987[4], *Domain names — Implementation and specification*

RFC 1630:1994, *Universal resource identifiers in WWW: A unifying syntax for the expression of names and addresses of objects on the network as used in the world-wide web*

FIPS-PUB 140-1:1993, *Security requirements for cryptographic modules*

3 Terms and definitions

For the purposes of this part of ISO 15782, the following terms and definitions apply.

3.1
attribute
characteristic of an entity

3.2
CA certificate
certificate whose subject is a Certification Authority (CA) and whose associated private key is used to sign certificates

3.3
certificate
public key and identity of an entity together with some other information, rendered unforgeable by signing the certificate information with the private key of the certifying authority that issued that public key certificate

3.4
certificate hold
suspension of the validity of a certificate

3.5
certificate policy
named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements

EXAMPLE A particular certificate policy might indicate the applicability of a type of certificate to the authentication of electronic data interchange transactions for the trading of goods within a given price range.

NOTE 1 The certificate policy should be used by the user of the certificate to decide whether or not to accept the binding between the subject (of the certificate) and the public key. A subset of the components in the certificate policy framework are given concrete values to define a certificate policy. The certificate policy is represented by a registered object identifier in the X.509, version 3 certificate. The object owner also registers a textual description of the policy and makes it available to the relying parties.

NOTE 2 The certificate policy object identifier can be included in the following extensions in the X.509, version 3 certificates: certificate policies, policy mappings and policy constraints. The object identifier(s) may appear in none, some, or all of these fields. These object identifiers may be the same (referring to the same certificate policy) or may be different (referring to different certificate policies).

3.6
Certificate Revocation List
CRL
list of revoked certificates

---

3) Obsoletes RFC 733; updated by RFC 987; updated by RFC 1327.

4) Obsoletes RFC 973; obsoleted by RFC 2136; obsoleted by RFC 2137; updated by RFC 1348; updated by RFC 1995; updated by RFC 1996; updated by RFC 2065; updated by RFC 2181; updated by RFC 2308.

3.7
certificate-using system
implementation of those functions defined in this part of ISO 15782 that are used by a certificate user

3.8
certification
process of creating a public key or attribute certificate for an entity

3.9
Certification Authority
CA
entity trusted by one or more entities to create assign and revoke or hold public key certificates

3.10
certification path
ordered sequence of certificates of entities which, together with the public key of the initial entity in the path, can be processed to obtain the public key of the final entity in the path

3.11
Certification Practice Statement
CPS
statement of the practices which a certification authority employs in issuing certificates

3.12
compromise
violation of the security of a system such that an unauthorized disclosure of sensitive information may have occurred

3.13
CRL distribution point
directory entry or other distribution source for Certificate Revocation Lists (CRLs)

NOTE A CRL distributed through a CRL distribution point may contain revocation entries for only a subset of the full set of certificates issued by one CA or may contain revocation entries for multiple CAs.

3.14
cross certification
process by which two CAs mutually certify each other’s public keys

See policy mapping (3.37).

3.15
cryptographic key
key
parameter that determines the operation of a cryptographic function

NOTE Cryptographic functions include:

- the transformation from plain text to cipher text and vice versa;
- synchronized generation of keying material;
- digital signature generation or validation.

3.16
cryptographic module
set of hardware, firmware, software or some combination thereof, that implements cryptographic logic, cryptographic processes, or both

3.17
cryptography
discipline which embodies principles, means and methods for the transformation of data in order to hide its information content, prevent its undetected modification, prevent its unauthorized use or a combination thereof

3.18
data integrity
property whereby data has not been altered or destroyed

3.19
delta-CRL
partial Certificate Revocation List (CRL) indicating only changes since a prior CRL issue

3.20
digital signature
signature
cryptographic transformation of data which, when associated with a data unit, provides the services of origin authentication and data integrity and may support signer non-repudiation

3.21
directory
repository
method for distributing or making available certificates or Certificate Revocation Lists (CRLs)

EXAMPLE A data base or an X.500 Directory.

3.22
distinguished name
globally unique name for an entity

NOTE Methods for determining global uniqueness are outside the scope of this part of ISO 15782. Note that an entity may be issued more than one certificate with the same distinguished name.

3.23
end certificate
last certificate considered in a certificate chain

3.24
end entity
certificate subject which uses its private key for purposes other than signing certificates

3.25
entity
legal or natural person who is a Certification Authority (CA), Registration Authority (RA) or end entity

3.26
financial message
communication containing information which has financial implications

3.27
intermediate certificates
certificate considered in a certificate chain other than the first or end certificate

3.28
key
see cryptographic key (3.15)

3.29
key agreement
method for negotiating a key value on-line without transferring the key, even in an encrypted form

EXAMPLE The Diffie-Hellman technique.

3.30
key pair
⟨public key cryptography⟩ public key and its corresponding private key

3.31
keying material
data, such as keys, certificates and initialization vectors, necessary to establish and maintain cryptographic keying relationships

3.32
key pair updating
re-certification or replacement of a CA's public/private key pair

3.33
message
data to be signed

3.34
module
see cryptographic module (3.16)

3.35
non-repudiation
service which provides proof beyond a reasonable doubt of the integrity and origin of data which can be validated by a third entity

NOTE The non-repudiation service protects against the signing entity falsely denying the action and may provide rebuttable presumption. It requires that appropriate processes and procedures (e.g., registration, audit trails, contractual arrangements, personnel, etc.) be in place.

3.36
optional
not required by this part of ISO 15782 or not required to meet an optional provision of this part of ISO 15782

NOTE Not to be confused with the ASN.1 key word "OPTIONAL".

3.37
policy mapping
recognition that, when a Certification Authority (CA) in one domain certifies a CA in another domain, a particular certificate policy in the second domain may be considered by the authority of the first domain to be equivalent (but not necessarily identical in all respects) to a particular certificate policy in the first domain

See cross certification (3.14).

3.38
policy qualifier
policy-dependent information that accompanies a certificate policy identifier in an X.509 certificate

3.39
private key
⟨asymmetric (public) key cryptosystem⟩ key of an entity's key pair which is known only by that entity

3.40
public key
⟨asymmetric (public) key cryptosystem⟩ key of an entity's key pair which is publicly known

3.41
Registration Authority
RA
entity that is responsible for identification and authentication of subjects of certificates, but is not a Certification Authority (CA) and hence does not sign or issue certificates

NOTE An RA may assist in the certificate application process, revocation process, or both. The RA does not need to be a separate body, but can be part of the CA.

3.42
relying party
recipient of a certificate who acts in reliance on that certificate, digital signatures verified using that certificate, or both

3.43
signature
see digital signature (3.20)

3.44
subject
entity whose public key is certified in a public key certificate

3.45
subject CA
Certification Authority (CA) that is certified by the issuing CA and hence complies with the certificate policy of the issuing CA

3.46
subject end entity
end entity that is the subject of a certificate

3.47
user
see relying party (3.42)

4 Abbreviations

The following abbreviations are used in this part of ISO 15782.

Abbreviation	**Meaning**
ASN.1	Abstract Syntax Notation
CA	Certification Authority
DIT	Directory Information Tree
CRL	Certificate Revocation List
ITU	International Telecommunication Union

NOTE 1 The notation used in this part of ISO 15782 is a variant of the X.509 notation for certificates, certification paths and related information.

NOTE 2 The use of a bold, sans serif font such as “**CertReqData**” or “**CRLEntry**” denotes the use of Abstract Syntax Notation (ASN.1). Where it makes sense to do so, the ASN.1 term is used in place of normal text.

5 Extensions

Version 3 certificates as defined in ISO/IEC 9594-8 provide a mechanism for CAs to append additional information about the:

— subject's public key;

— issuer's public key;

— issuer's CRLs.

This additional information is encoded in the form of extensions to certificates.

These extensions are specified in the following areas:

a) *Key and policy information*: These certificate and CRL extensions convey additional information about the keys involved, including key identifiers for subject and issuer keys, indicators of intended or restricted key usage and indicators of certificate policy.

b) *Certificate subject and issuer attributes*: These certificate and CRL extensions support alternative names, of various name forms, for a certificate subject, a certificate issuer, or a CRL issuer. These extensions can also convey additional attribute information about the certificate subject, to assist a relying party in being confident that the certificate subject is a particular person or entity.

c) *Certification path constraints*: These certificate extensions allow constraint specifications to be included in CA certificates, i.e. certificates for CAs issued by other CAs, to facilitate the automated processing of certification paths when multiple certificate policies are involved. Multiple certificate policies arise when policies vary for different applications in an environment or when interoperation with external environments occurs. The constraints may restrict the types of certificates that can be issued by the subject CA or that may occur subsequently in a certification path.

d) *Basic CRL extensions*: These CRL extensions allow a CRL to include indications of revocation reason, to provide for temporary suspension of a certificate and to include CRL-issue sequence numbers to allow relying parties to detect missing CRLs in a sequence from one CRL issuer.

e) *CRL distribution points and delta-CRLs*: These certificate and CRL extensions allow the complete set of revocation information from one CA to be partitioned into separate CRLs and allow revocation information from multiple CAs to be combined in one CRL. These extensions also support the use of partial CRLs indicating only changes since an earlier CRL issue.

Inclusion of any extension in a certificate or CRL is at the option of the authority issuing that certificate or CRL.

In a certificate or CRL, an extension is flagged as being either critical or non-critical. If an extension is flagged critical and a certificate-using system does not recognize the extension type or does not implement the semantics of the extension, then that system shall consider the certificate invalid. If an extension is flagged non-critical, a certificate-using system that does not recognize or implement that extension type may process the remainder of the certificate ignoring the extension. Extension type definitions in this part of ISO 15782 indicate if the extension is always critical, always non-critical, or if criticality can be decided by the certificate or CRL issuer. The reason for requiring some extensions to be always non-critical is to allow certificate-using implementations which do not need to use such extensions to omit support for them without jeopardizing the ability to interoperate with all certification authorities.

These extensions provide a variety of methods to increase the amount of information that the certificate conveys to facilitate automated certificate processing. The extensions are intended to allow explicit management of trust and policies corresponding to the differing needs within an organization and certification across hierarchies.

A certificate-using system may require certain non-critical extensions to be present in a certificate in order for that certificate to be considered acceptable. The need for inclusion of such extensions may be:

a) implied by local policy rules of the relying party, or

b) a CA policy rule indicated to the certificate-using system by inclusion of a particular certificate policy identifier in the certificate policies extension with that extension being flagged critical.

There shall be no more than one instance of each extension type in any certificate, CRL, or CRL entry, respectively.

Subclause 10.4 also defines matching rules to facilitate the selection of certificates or CRLs with specific characteristics from multiple-valued attributes holding multiple certificates or CRLs.

Implementers are cautioned that the level of complexity inherent in

— policy mappings extension (see 6.2.8),

— name constraints extension (see 8.2.3),

— policy constraints extension (see 8.2.4) and

— matching rules (see 10.4)

is significantly greater than that of the rest of this part of ISO 15782. The decision to implement these extensions should be based on business and risk after consideration of alternative architectures or alternative methods.

6 Key and policy information

6.1 Requirements

The following requirements relate to key and policy information:

a) CA key pair updating can occur at regular intervals or in special circumstances. There is a need for a certificate extension to convey an identifier of the public key to be used to verify the certificate signature. A certificate-using system can use such identifiers in finding the correct CA certificate for validating the certificate issuer's public key.

b) In general, a certificate subject has different public keys and, correspondingly, different certificates for different purposes, e.g. digital signature, encipherment and key agreement. A certificate extension is needed to assist a relying party in selecting the correct certificate for a given subject for a particular purpose or to allow a CA to stipulate that a certified key may only be used for a particular purpose.

c) Subject key pair updating can occur at regular intervals or in special circumstances. There is a need for a certificate extension to convey an identifier to distinguish between different public keys for the same subject used at different points in time. A certificate-using system can use such identifiers in finding the correct certificate.

d) With digital signature keys, the usage period for the signing private key is typically shorter than that for the verifying public key. In the event of a private key compromise, the period of exposure can be limited if the signature verifier knows the legitimate use period for the private key. There is therefore a requirement to be able to indicate the usage period of the private key in a certificate.

e) Because certificates may be used in environments where multiple certificate policies apply, provision needs to be made for including certificate policy information in certificates.

f) When cross-certifying from one organization to another, it can sometimes be agreed that certain of the two organizations' policies can be considered equivalent. A CA certificate needs to allow the certificate issuer to indicate that one of its own certificate policies is equivalent to another certificate policy in the domain of the subject CA. This is known as policy mapping.

g) A user of an encipherment or digital signature system which uses certificates defined in ISO 15782-1 and this part of ISO 15782 needs to be able to determine in advance the algorithms supported by other users.

6.2 Certificate and CRL extensions

6.2.1 Overview

The following extensions are defined:

a) authority key identifier;

b) subject key identifier;

c) key usage;

d) extended key usage

e) private key usage period;

f) certificate policies;

g) policy mappings.

These extensions shall be used only as certificate extensions, except for authority key identifier which may also be used as a CRL extension. Unless noted otherwise, these extensions may be used in both CA certificates and end entity certificates.

In addition, a directory attribute is defined to support the selection of an algorithm for use when communicating with a remote end entity using certificates as defined in ISO 15782-1 and in this part of ISO 15782.

6.2.2 Authority key identifier extension

This extension, which may be used as either a certificate extension or CRL extension, identifies the public key to be used to verify the signature on this certificate or CRL. It enables distinct keys used by the same CA to be distinguished (e.g. as key updating occurs). This extension is defined as follows:

```
authorityKeyIdentifier EXTENSION ::= {
    SYNTAX          AuthorityKeyIdentifier
    IDENTIFIED BY   id-ce-authorityKeyIdentifier }

AuthorityKeyIdentifier ::= SEQUENCE {
    keyIdentifier               [0] KeyIdentifier        OPTIONAL,
    authorityCertIssuer         [1] GeneralNames            OPTIONAL,
    authorityCertSerialNumber   [2] CertificateSerialNumber   OPTIONAL
}
( WITH COMPONENTS    {..., authorityCertIssuer PRESENT,
    authorityCertSerialNumber   PRESENT} |
        WITH COMPONENTS         {..., authorityCertIssuer ABSENT,
    authorityCertSerialNumber    ABSENT} )

KeyIdentifier ::= OCTET STRING
```

The key may be identified by:

— an explicit key identifier in the **keyIdentifier** component;

— by identification of a certificate for the key (giving certificate issuer in the **authorityCertIssuer** component and certificate serial number in the **authorityCertSerialNumber** component); or

— by both explicit key identifier and identification of a certificate for the key.

If both forms of identification are used, then the certificate or CRL issuer shall ensure they are consistent.

A key identifier shall be unique with respect to all key identifiers for the issuing authority for the certificate or CRL containing the extension. An implementation which supports this extension is not required to be able to process all name forms in the **authorityCertIssuer** component. See 7.2.2 for details of the **GeneralNames** type.

Certification authorities shall assign certificate serial numbers such that every (issuer, certificate serial number) pair uniquely identifies a single certificate.

This extension is always non-critical.

6.2.3 Subject key identifier extension

This extension identifies the public key being certified. It enables distinct keys used by the same subject to be differentiated (e.g. as key updating occurs). This extension is defined as follows:

```
subjectKeyIdentifier EXTENSION ::= {
    SYNTAX          SubjectKeyIdentifier
    IDENTIFIED BY   id-ce-subjectKeyIdentifier }

SubjectKeyIdentifier ::= KeyIdentifier
```

A key identifier shall be unique with respect to all key identifiers for the subject with which it is used. This extension is always non-critical.

6.2.4 Key usage extension

This extension, which indicates the purpose for which the certified public key is used, is defined as follows:

```
keyUsage EXTENSION ::= {
    SYNTAX          KeyUsage
    IDENTIFIED BY   id-ce-keyUsage }

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** field are as follows:

a) **digitalSignature**: asserted when the subject public key is used with a digital signature mechanism to support security services other than non-repudiation (bit 1), certificate signing (bit 5), or revocation information signing (bit 6). Digital signature mechanisms are often used for entity authentication and data origin authentication with integrity;

b) **nonRepudiation**: for verifying a digital signature applied to an object to support the delivery of the non-repudiation service [excluding certificate or CRL signing as in f) or g)]$^{5)}$;

c) **keyEncipherment**: for enciphering keys or other security information, e.g. for key transport;

CAs shall set the **keyEncipherment** bit when the public key is used for enciphering keys or other security information.

d) **dataEncipherment**: for enciphering user data, but not keys or other security information as in c) above;

CAs shall set the dataEncipherment bit when the public key is used for enciphering user data, but not keys or other security information.

e) **keyAgreement**: for use as a public key agreement key;

CAs shall set the **keyAgreement** bit when the public key is used as a key agreement key.

f) **keyCertSign**: for verifying a CA's signature on certificates;

CAs shall set the **keyCertSign** bit when the public key may be used to sign certificates. This bit shall only be set in CA certificates.

g) **cRLSign**: for verifying a CA's signature on CRLs.

CAs must set the **cRLSign** bit when the public key may be used to sign CRLs.

h) The **encipherOnly** and **decipherOnly** key usages are intended to provide support for key agreement schemes where separate shared secret keys are used in each direction of communication. In such a scheme, a user has more than one set of key pairs and bits 7 (**encipherOnly**) and 8 (**decipherOnly**) are used to distinguish between the two types. The originator of a message would use the recipient's public key certificate with bits 4 (**keyAgreement**) and 7 (**encipherOnly**) to create a key encryption key. The recipient would use the originator's certificate with bits 4 (**keyAgreement**) and 8 (**decipherOnly**) to create the key encryption key. Typically the originator would pass his own certificate with bits 4 and 8 along with the message. Financial systems are not required to implement a key management scheme where the symmetric keys used in each direction of communication are derived from separate public key pairs. The **encipherOnly** and **decipherOnly** shall not both be set to **TRUE**.

CAs shall include this extension in all CA and end entity certificates and indicate that the extension is critical by setting the criticality flag to "true".

CAs shall set the key usage bits according to the valid key usage combinations as illustrated and described in ISO 15782-1:—, subclause 9.1.3 and Table 8.

6.2.5 Extended key usage extension

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. This extension is defined as follows:

```
extKeyUsage EXTENSION ::= {
    SYNTAX           SEQUENCE SIZE (1..MAX) OF KeyPurposeId
    IDENTIFIED BY    id-ce-extKeyUsage }

KeyPurposeId ::= OBJECT IDENTIFIER
```

5) Non repudiation is only a business issue, whereas digital signature and encryption are technical issues, consequently the key usage bits in a certificate *should not* address non repudiation.

Key purposes may be defined by any organization with a need. Object identifiers used to identify key purposes shall be assigned in accordance with ISO/IEC 9834-1 | CCITT Rec. X.660.

This extension may, at the option of the certificate issuer, be either critical or non-critical.

If the extension is flagged critical, then the certificate shall be used only for one of the purposes indicated.

If the extension is flagged non-critical, then it indicates the intended purpose or purposes of the key and may be used in finding the correct key/certificate of an entity that has multiple keys/certificates. It is an advisory extension and does not imply that usage of the key is restricted by the certification authority to the purpose indicated. (Using applications may nevertheless require that a particular purpose be indicated in order for the certificate to be acceptable to that application.)

If a certificate contains both a critical key usage extension and a critical extended key usage extension, then both extensions shall be processed independently and the certificate shall only be used for a purpose consistent with both extensions. If there is no purpose consistent with both extensions, then the certificate shall not be used for any purpose.

6.2.6 Private key usage period extension

This extension indicates the period of use of the private key corresponding to the certified public key. It is applicable only for digital signature keys. This extension is defined as follows:

```
privateKeyUsagePeriod EXTENSION ::= {
    SYNTAX          PrivateKeyUsagePeriod
    IDENTIFIED BY   id-ce-privateKeyUsagePeriod }

PrivateKeyUsagePeriod ::= SEQUENCE {
    notBefore    [0]   GeneralizedTime OPTIONAL,
    notAfter     [1]   GeneralizedTime OPTIONAL }
        ( WITH COMPONENTS    {..., notBefore PRESENT} |
        WITH COMPONENTS           {..., notAfter PRESENT} )
```

The **notBefore** component indicates the earliest date and time at which the private key could be used for signing. If the **notBefore** component is not present, then no information is provided as to when the period of valid use of the private key commences. The **notAfter** component indicates the latest date and time at which the private key could be used for signing. If the **notAfter** component is not present, then no information is provided as to when the period of valid use of the private key concludes.

The period of valid use of the private key may be different from the certified validity of the public key as indicated by the certificate validity period. With digital signature keys, the usage period for the signing private key is typically shorter than that for the verifying public key.

This extension is always non-critical.

NOTE 1 While this extension is not strictly prohibited from being used with other than signature certificates, it is not logical to use this extension with a certificate other than a signature certificate.

NOTE 2 For digital signature, the private key is used to generate digital signatures. If a validity period is given for the signature private key, then the key may only be used to generate signatures within the validity period. For key management, the private key is used to decrypt encrypted items. Limiting a key management private key would inhibit decryption of encrypted data, hence, it would be unwise to assign a validity period to a key management private key because that would inhibit decryption of past data after the validity period expires.

6.2.7 Certificate policies extension

This extension lists certificate policies, recognized by the issuing CA, that apply to the certificate, together with optional qualifier information pertaining to these certificate policies. Typically, different certificate policies will relate to different applications which may use the certified key. This extension is defined as follows:

```
certificatePolicies EXTENSION ::= {
    SYNTAX          CertificatePoliciesSyntax
        IDENTIFIED BY    id-ce-certificatePolicies }

CertificatePoliciesSyntax ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation

PolicyInformation ::= SEQUENCE {
    policyIdentifier        CertPolicyId,
    policyQualifiers        SEQUENCE SIZE (1..MAX) OF PolicyQualifierInfo        OPTIONAL }

CertPolicyId ::= OBJECT IDENTIFIER

PolicyQualifierInfo ::= SEQUENCE {
    policyQualifierId CERT-POLICY-QUALIFIER.&id
        ({SupportedPolicyQualifiers}),
    qualifier CERT-POLICY-QUALIFIER.&Qualifier
        ({SupportedPolicyQualifiers}{@policyQualifierId})
        OPTIONAL }

SupportedPolicyQualifiers CERT-POLICY-QUALIFIER ::= { ... }
```

A value of the **PolicyInformation** type identifies and conveys qualifier information for one certificate policy. The component **policyIdentifier** contains an identifier of a certificate policy and the component **policyQualifiers** contains policy qualifier values for that element.

This extension may, at the option of the certificate issuer, be either critical or non-critical.

If the extension is flagged critical, it indicates that the certificate shall only be used for the purpose and in accordance with the rules implied by one of the indicated certificate policies. The rules of a particular policy may require the certificate-using system to process the qualifier value in a particular way.

If the extension is flagged non-critical, use of this extension does not necessarily constrain use of the certificate to the policies listed. However, a relying party may require a particular policy to be present in order to use the certificate (see 8.3). Policy qualifiers may, at the option of the relying party, be processed or ignored.

Certificate policies and certificate policy qualifier types may be defined by any organization with a need. Object identifiers used to identify certificate policies and certificate policy qualifier types shall be assigned in accordance with ISO/IEC 9834-1 | CCITT Rec. X.660. The following ASN.1 object class is used in defining certificate policy qualifier types:

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

A definition of a policy qualifier type shall include:

— a statement of the semantics of the possible values;

— an indication of whether the qualifier identifier may appear in a certificate policies extension without an accompanying value and, if so, the implied semantics in such a case.

A qualifier may be specified as having any ASN.1 type. When the qualifier is anticipated to be used primarily with applications that do not have ASN.1 decoding functions, it is recommended that the type **OCTET STRING** be specified. The ASN.1 **OCTET STRING** value can then convey a qualifier value encoded according to any convention specified by the policy element defining organization.

6.2.8 Policy mappings extension

This extension, which shall be used in CA certificates only, allows a certificate issuer to indicate that, for the purposes of the user of a certification path containing this certificate, one of the issuer's certificate policies can be considered equivalent to a different certificate policy used in the subject CA's domain. This extension is defined as follows:

```
policyMappings EXTENSION ::= {
      SYNTAX            PolicyMappingsSyntax
      IDENTIFIED BY     id-ce-policyMappings }

PolicyMappingsSyntax ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
      issuerDomainPolicy        CertPolicyId,
      subjectDomainPolicy       CertPolicyId }
```

The **issuerDomainPolicy** component indicates a certificate policy that is recognized in the issuing CA's domain and that can be considered equivalent to the certificate policy indicated in the **subjectDomainPolicy** component that is recognized in the subject CA's domain.

This extension is always non-critical.

NOTE 1 Policy mapping implies significant administrative overheads and the involvement of suitably diligent and authorized personnel in related decision-making. In general, it is preferable to agree upon more global use of common policies than it is to apply policy mapping. See annex A.

NOTE 2 It is anticipated that policy mapping will be practical only in limited environments in which policy statements are very simple.

6.2.9 Supported algorithms attribute

A Directory attribute is defined to support the selection of an algorithm for use when communicating with a remote end entity using certificates. The following ASN.1 defines this (multi-valued) attribute:

```
supportedAlgorithms ATTRIBUTE ::= {
      WITH SYNTAX                 SupportedAlgorithm
      EQUALITY MATCHING RULE      algorithmIdentifierMatch
      ID                          id-at-supportedAlgorithms }

SupportedAlgorithm ::= SEQUENCE {
      algorithmIdentifier         AlgorithmIdentifier,
      intendedUsage               [0] KeyUsage OPTIONAL,
      intendedCertificatePolicies [1] CertificatePoliciesSyntax OPTIONAL }
```

Each value of the multi-valued attribute shall have a distinct **algorithmIdentifier** value. The value of the **intendedUsage** component provides an indication of the intended usage of the algorithm. The value of the **intendedCertificatePolicies** component identifies the certificate policies and, optionally, certificate policy qualifiers with which the identified algorithm may be used.

7 Certificate subject and certificate issuer attributes

7.1 Requirements

The following requirements relate to certificate subject and certificate issuer attributes:

a) Certificates need to be usable by applications that employ a variety of name forms, including:

— internet electronic mail names;

— internet domain names;

— X.400 originator/recipient addresses;

— EDI party names.

It is therefore necessary to be able to securely associate multiple names of a variety of name forms with a certificate subject or a certificate or CRL issuer.

b) A relying party may need to securely know certain identifying information about a subject in order to have confidence that the subject is indeed the entity intended. For example, information such as postal address, position in a corporation, or a picture image may be required. Such information may be conveniently represented as directory attributes, but these attributes are not necessarily part of the distinguished name. A certificate extension is therefore needed for conveying additional directory attributes beyond those in the distinguished name.

7.2 Certificate and CRL extensions

7.2.1 Overview

The following extensions are defined:

a) subject alternative name;

b) issuer alternative name;

c) subject directory attributes.

These extensions shall be used only as certificate extensions, except for issuer alternative name which may also be used as a CRL extension. As certificate extensions, they may be present in CA certificates or end entity certificates.

7.2.2 Subject alternative name extension

This extension contains one or more alternative names, using any of a variety of name forms, for the entity that is bound by the CA to the certified public key. These alternative names must be associated with the same entity as subject name during the registration process. This extension is defined as follows:

```
subjectAltName EXTENSION ::= {
    SYNTAX          GeneralNames
    IDENTIFIED BY   id-ce-subjectAltName }

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

GeneralName ::= CHOICE {
    otherName                  [0]  INSTANCE OF OTHER-NAME,
    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 }

OTHER-NAME ::= TYPE-IDENTIFIER

EDIPartyName ::= SEQUENCE {
    nameAssigner   [0]  DirectoryString {ub-name} OPTIONAL,
    partyName      [1]  DirectoryString {ub-name} }
```

The values in the alternatives of the **GeneralName** type are names of various forms as follows:

— **otherName** is a name of any form defined as an instance of the **OTHER-NAME** information object class;

— **rfc822Name** is an Internet electronic mail address defined in accordance with Internet RFC 822;

— **dNSName** is an Internet domain name defined in accordance with Internet RFC 1035;

— **x400Address** is an O/R address defined in accordance with ISO/IEC 10021-4| ITU-T Rec. X.411;

— **directoryName** is a directory name defined in accordance with ISO/IEC 9594-2| ITU-T Rec. X.501;

— **ediPartyName** is a name of a form agreed between communicating Electronic Data Interchange partners; the **nameAssigner** component identifies an authority that assigns unique values of names in the **partyName** component;

— **uniformResourceIdentifier** is a Uniform Resource Identifier for the World-Wide Web defined in accordance with Internet RFC 1630;

— **iPAddress** is an Internet Protocol address defined in accordance with Internet RFC 791, represented as a binary string.

— **registeredID** is an identifier of any registered object assigned in accordance with ISO/IEC 9834-1| CCITT Rec. X.660.

For every name form used in the **GeneralName** type, there shall be a name registration system that ensures that any name used unambiguously identifies one entity to both certificate issuer and relying parties.

This extension may, at the option of the certificate issuer, be either critical or non-critical. An implementation which supports this extension is not required to be able to process all name forms. If the extension is flagged critical, at least one of the name forms that is present shall be recognized and processed, otherwise the certificate shall be considered invalid. Apart from the preceding restriction, a certificate-using system is permitted to ignore any name with an unrecognized or unsupported name form. It is recommended that, provided the subject field of the certificate contains a directory name that unambiguously identifies the subject, this extension be flagged non-critical.

If this extension is present and is flagged critical, the subject field of the certificate may contain a null name (e.g. a sequence of zero relative distinguished names) in which case the subject is identified only by the name or names in this extension.

NOTE Use of the **TYPE-IDENTIFIER** class is described in annex A of ISO/IEC 8824-2:1998.

7.2.3 Issuer alternative name extension

This extension contains one or more alternative names, using any of a variety of name forms, for the certificate or CRL issuer. This extension is defined as follows:

```
issuerAltName EXTENSION ::= {
    SYNTAX          GeneralNames
    IDENTIFIED BY   id-ce-issuerAltName }
```

This extension may, at the option of the certificate or CRL issuer, be either critical or non-critical. An implementation which supports this extension is not required to be able to process all name forms. If the extension is flagged critical, at least one of the name forms that is present shall be recognized and processed, otherwise the certificate or CRL shall be considered invalid. Apart from the preceding restriction, a certificate-using system is permitted to ignore any name with an unrecognized or unsupported name form. It is recommended that, provided the issuer field of the certificate or CRL contains a directory name that unambiguously identifies the issuing authority, this extension be flagged non-critical.

If this extension is present and is flagged critical, the **issuer** field of the certificate or CRL may contain a null name (e.g. a sequence of zero relative distinguished names) in which case the issuer is identified only by the name or names in this extension.

7.2.4 Subject directory attributes extension

This extension conveys any desired directory attribute values for the subject of the certificate. This extension is defined as follows:

```
subjectDirectoryAttributes EXTENSION ::= {
    SYNTAX           AttributesSyntax
    IDENTIFIED BY    id-ce-subjectDirectoryAttributes }
```

```
AttributesSyntax ::= SEQUENCE SIZE (1..MAX) OF Attribute
```

This extension is always non-critical.

8 Certification path constraints

8.1 Requirements

For certification path processing:

a) Basic constraints:

   1) To protect against end-entities establishing themselves as CAs without authorization, end entity certificates shall be distinguishable from CA certificates.

   2) It shall be possible for a CA to limit the length of a subsequent chain resulting from a certified subject CA.

b) Name constraints:

   1) A CA shall be able to specify constraints which allow a relying party to verify that CAs in a certification path are not violating their trust by issuing certificates to subjects in an inappropriate name space.

   2) A CA shall be able to stipulate which CAs in other domains it trusts and for which purposes.

c) Policy constraints:

   1) Certification paths may operate in environments in which multiple certificate policies are recognized.

   2) Chaining through multiple policy domains should be supported.

   3) A CA shall be able to inhibit the use of policy mapping and to require that explicit certificate policy identifiers be present in subsequent certificates in a certification path.

In any certificate-using system, processing of a certification path requires an appropriate level of assurance. This part of ISO 15782 defines functions that may be used in implementations that are required to conform to specific assurance statements. For example, an assurance requirement could state that certification path processing shall be protected from subversion of the process (such as software-tampering or data modification).

A cryptographic module provides security either:

— in accordance with FIPS 140-1, except for requirements for cryptographic algorithms; or

— by conforming to the requirements of a protection profile approved in an ISO or national standard for *very high* risk financial applications.

NOTE These levels are not to be confused with Common Criteria Evaluation Assurance Levels (EALS) or Security Functional Levels (FCNs), or with levels of trust that may be defined in a CA’s Certificate Policy/CPS.

Four levels of cryptographic module security are defined as follows:

— Level 1 cryptographic module

Security level 1 provides the lowest level of security. It specifies basic security requirements for a cryptographic module, but it differs from the higher levels in several respects. No physical security mechanisms are required in the module beyond the requirement for production-grade equipment.

Level 1 allows software cryptographic functions to be performed in a general purpose personal computer (PC). Such implementations are often appropriate in low-level security applications. The implementation of PC cryptographic software may be more cost-effective than hardware-based mechanisms. This will enable organizations to avoid the situation that exists today whereby the decision is often made not to cryptographically protect data because hardware is considered too expensive.

— Level 2 cryptographic module (tamper evident)

Level 2 provides physical security by including requirements for tamper evident coatings or seals, or for pick-resistant locks. Tamper evident coatings or seals would be placed on a cryptographic module so that the coating or seal would have to be broken in order to attain physical access to the plaintext cryptographic keys and other critical security parameters within the module. Pick-resistant locks would be placed on covers or doors to protect against unauthorized physical access. These requirements provide a low-cost means for physical security and avoid the cost of the higher level of protection involving hard opaque coatings or significantly more expensive tamper detection and zeroization circuitry.

— Level 3 cryptographic module (tamper protected)

Level 3 attempts to prevent the intruder from gaining access to critical security parameters held within the module. For example, a multiple chip embedded module shall be contained in a strong enclosure and if a cover is removed or a door is opened, the critical security parameters are zeroized. As another example, a module shall be enclosed in a hard, opaque potting material to deter access to the contents.

— Level 4 cryptographic module (tamper enveloped)

Level 4 physical security provides an envelope of protection around the cryptographic module. Whereas the tamper detection circuits of lower level modules may be bypassed, the intent of level 4 protection is to detect a penetration of the device from any direction. For example, if one attempts to cut through the enclosure of the cryptographic module, the attempt should be detected and all critical security parameters should be zeroized. Level 4 devices are particularly useful for operation in a physically unprotected environment where an intruder could possibly tamper with the device.

The level of assurance should be commensurate with business risk. For example:

— processing internal to an appropriate cryptographic module may be required for public keys used to validate high value funds transfer; whereas

— processing in software may be appropriate for home banking balance inquiries.

Consequently, certification path processing functions should be suitable for implementation in hardware cryptographic modules or cryptographic tokens as one option.

It should be possible to implement certification path processing without depending upon the use of trusted local databases of policy-description information. (Some trusted local information — an initial public key, at least — is needed for certification path processing but the amount of such information should be minimized.)

8.2 Certificate extensions

8.2.1 Overview

The following extensions are defined:

a) basic constraints;

b) name constraints;

c) policy constraints.

These extensions shall only be used as certificate extensions. Name constraints and policy constraints shall be used only in CA certificates; basic constraints shall also be used in end entity certificates. Examples of the use of these extensions are given in annex A.

8.2.2 Basic constraints extension

This extension indicates if the subject may act as a CA, with the certified public key being used to verify certificate signatures. If so, a certification path length constraint may also be specified. This extension is defined as follows:

```
basicConstraints EXTENSION ::= {
    SYNTAX          BasicConstraintsSyntax
    IDENTIFIED BY   id-ce-basicConstraints }

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

The **cA** component indicates if the certified public key may be used to verify certificate signatures.

The **pathLenConstraint** component shall be present only if **cA** is set to true. It gives the maximum number of CA certificates that may follow this certificate in a certification path. Value 0 indicates that the subject of this certificate may issue certificates only to end-entities and not to further CAs. If no **pathLenConstraint** extension appears in any certificate of a certification path, there is no limit to the allowed length of the certification path.

This extension shall be flagged critical. Otherwise an entity which is not authorized to be a CA may issue certificates and a certificate-using system may unwittingly use such a certificate.

If this extension is present and is flagged critical then:

— if the value of **cA** is not set to true, then the certified public key shall not be used to verify a certificate signature;

— if the value of **cA** is set to true and **pathLenConstraint** is present, then the certificate-using system shall check that the certification path being processed is consistent with the value of **pathLenConstraint**.

To constrain a certificate subject to being only an end entity, i.e. not a CA, the issuer shall include this extension containing only an empty **SEQUENCE** value.

8.2.3 Name constraints extension

All name forms that appear in a certificate shall be validated against all applicable constraints. This includes the **subject** and **subjectAltName** extensions.

This extension, which shall be used only in a CA certificate, indicates a name space within which all subject names in subsequent certificates in a certification path shall be located and is defined as follows:

```
nameConstraints EXTENSION ::= {
    SYNTAX          NameConstraintsSyntax
    IDENTIFIED BY   id-ce-nameConstraints }

NameConstraintsSyntax ::= 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 CA shall include the **permittedSubtrees** and **excludedSubtrees** components. Each specifies one or more naming subtrees, each defined by the name of the root of the subtree and, optionally, within that subtree, an area that is bounded by upper and/or lower levels.

— Of all the certificates issued by the subject CA and subsequent CAs in the certification path, only those certificates with subject names within these subtrees are acceptable.

— Any certificate issued by the subject CA or subsequent CAs in the certification path that has a subject name within these subtrees is unacceptable.

— If both **permittedSubtrees** and **excludedSubtrees** are present and the name spaces overlap, the exclusion statement takes precedence.

Of the name forms available through the **GeneralName** type, only those name forms that have a well-defined hierarchical structure may be used in these extensions. The **directoryName** name form satisfies this requirement; when using this name form, a naming subtree corresponds to a Directory Information Tree (DIT) subtree.

Conformant implementations are not required to recognize all possible name forms.

— If the extension is flagged critical and a certificate-using implementation does not recognize a name form used in any **base** component, the certificate shall be handled as if an unrecognized critical extension had been encountered.

— If the extension is flagged non-critical and a certificate-using implementation does not recognize a name form used in any **base** component, then that subtree specification may be ignored.

When validating certificate subject names for consistency with a name constraint, names in non-critical subject alternative name extensions shall be processed, not ignored.

The **minimum** field specifies the upper bound of the area within the subtree. All names whose final name component is above the level specified are not contained within the area. A value of **minimum** equal to zero (the default) corresponds to the base, i.e. the top node of the subtree. For example, if **minimum** is set to one, then the naming subtree excludes the base node but includes subordinate nodes.

The **maximum** field specifies the lower bound of the area within the subtree. All names whose last component is below the level specified are not contained within the area. A value of **maximum** of zero corresponds to the base, i.e. the top of the subtree. An absent **maximum** component indicates that no lower limit should be imposed on the area within the subtree. For example, if **maximum** is set to one, then the naming subtree excludes all nodes except the subtree base and its immediate subordinates.

This extension may, at the option of the certificate issuer, be either critical or non-critical. It should be flagged critical, to allow a relying party to check that subsequent certificates in a certification path are located in the name space intended by the issuing CA.

If this extension is present and is flagged critical, then a certificate-using system shall check that the certification path being processed is consistent with the value in this extension.

8.2.4 Policy constraints extension

This extension specifies constraints which may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path and is defined as follows:

```
policyConstraints EXTENSION ::= {
    SYNTAX          PolicyConstraintsSyntax
    IDENTIFIED BY   id-ce-policyConstraints }

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

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

If the **requireExplicitPolicy** component is present, all certificates starting from that issued by a nominated CA in the certification path until the end of the certification path shall contain an acceptable policy identifier in the certificate policies extension. An acceptable policy identifier is:

— the identifier of the certificate policy required by the user of the certification path; or

— the identifier of a policy which has been declared equivalent to it through policy mapping.

The nominated CA is either the subject CA of the certificate containing this extension (if the value of **requireExplicitPolicy** is 0) or a CA which is the subject of a subsequent certificate in the certification path (as indicated by a non-zero value).

If the **inhibitPolicyMapping** component is present, policy mapping is not permitted in all certificates starting from a nominated CA in the certification path until the end of the certification path.

A value of type **SkipCerts** indicates the number of certificates in the certification path to skip before a constraint becomes effective.

It should be flagged critical to ensure that the certificate is used in a manner consistent with its policy.

8.3 Certification path processing procedure

The certificate policies, basic constraints, name constraints and policy constraints extensions have been designed to facilitate automated, self-contained implementation of certification path processing logic.

Certification path processing is carried out in a system which needs to use the public key of a remote end entity, e.g. a system which is verifying a digital signature generated by an entity.

The following is an outline of a procedure for validating certification paths. An implementation shall be functionally equivalent to the external behaviour resulting from this procedure.

The inputs to the certification path processing procedure are:

a) a set of certificates comprising a certification path;

b) a trusted public key value or key identifier (if the key is stored internally to the certification path processing module), for use in verifying the first certificate in the certification path;

c) an *initial-policy-set* comprising one or more certificate policy identifiers, indicating that any one of these policies would be acceptable to the relying party for the purposes of certification path processing; this input can also take the special value *any-policy*;

d) an *initial-explicit-policy* indicator value, which indicates if an acceptable policy identifier needs to explicitly appear in the certificate policies extension of all certificates in the path;

e) an *initial-policy-mapping-inhibit* indicator value, which indicates if policy mapping is forbidden in the certification path;

f) the current date/time (if not available internally to the certification path processing module).

The values of c), d) and e) will depend upon the policy requirements of the user-application combination that needs to use the certified end entity public key.

The outputs of the procedure are:

a) an indication of success or failure of certification path validation;

   1) if validation failed, a diagnostic code indicating the reason for failure;

   2) if validation was successful, a set of policies constrained by the CAs in the certification path, together with all qualifiers for these policies encountered in the certification path, or the special value *any-policy*. Unless *any-policy* is returned, the relying party shall only use the certificate in accordance with one of the identified policies and shall process all qualifiers for that policy present in the certification path.

   3) if validation was successful and the step 2) returned the value *any-policy*, the set of all policy element qualifiers encountered in the certification path.

b) details of any policy mapping that occurred in processing the certification path.

If validation is successful, the certificate-using system may still take the business decision not to use the certificate as a result of values of policy qualifiers or other information in the certificate.

The procedure makes use of the following set of state variables:

a) *user-constrained-policy-set*: A set of certificate policy identifiers comprising the policies currently considered acceptable to the relying party; this state variable can also take the special value *any-policy*;

b) *authority-constrained-policy-set*: A set of certificate policy identifiers comprising the set of policies currently considered acceptable to the CAs in the certification path; this state variable can also take the special value *any-policy*;

c) *permitted-subtrees*: A set of subtree specifications defining subtrees within which all subject names in subsequent certificates in the certification path shall fall, or may take the special value *unbounded*;

d) *excluded-subtrees*: A (possibly empty) set of subtree specifications (each comprising a subtree base name and maximum and minimum level indicators) defining subtrees within which no subject name in a subsequent certificate in the certification path may fall;

e) *explicit-policy-indicator*: Indicates if an acceptable policy needs to be explicitly identified in every certificate;

f) *policy-mapping-inhibit-indicator*: Indicates if policy mapping is inhibited;

g) *pending-constraints*: Details of explicit-policy and/or inhibit-policy-mapping constraints which have been stipulated but are yet to take effect. There are two one-bit indicators called *explicit-policy-pending* and *policy-*

mapping-inhibit-pending together with, for each, an integer called *skip-certificates* which gives the number of certificates yet to skip before the constraint takes effect.

The procedure involves an initialization step, followed by a series of certificate-processing steps. The initialization step comprises:

a) Initialize the user-constrained-policy-set variable to the value of initial-policy-set;

b) Initialize the authority-constrained-policy-set variable to the value any-policy;

c) Initialize the permitted-subtrees variable to *unbounded*;

d) Initialize the excluded-subtrees variable to an empty set;

e) Initialize the explicit-policy-indicator to the initial-explicit-policy value;

f) Initialize the policy-mapping-inhibit-indicator to the initial-policy-mapping-inhibit value;

g) Initialize the two pending-constraints indicators to unset.

Each certificate is then processed in turn, starting with the certificate signed using the input trusted public key. The last certificate is considered to be the *end certificate*; any other certificates are considered to be *intermediate certificates*.

The following checks are applied to a certificate:

a) Check that:

   1) the signature verifies,

   2) dates are valid,

   3) the certificate subject and certificate issuer names chain correctly,

   4) the certificate has not been revoked.

b) For an intermediate certificate:

   1) the basic constraints extension shall be present in the certificate and the **cA** component shall be set to true.

   2) If the **pathLenConstraint** component is present, check that the current certification path does not violate that constraint.

c) For an end certificate, the basic constraint extension shall be present in the certificate and the **cA** component shall be set to **FALSE**.

d) If *explicit-policy-indicator* is set and *user-constrained-policy-set* is not set to *any-policy*, check that the certificate policies extension is present and that at least one member of *user-constrained-policy-set* appears in the certificate policies extension.

e) If the certificate policies extension is present and is flagged critical, compute the intersection of the policies in that extension and the *authority-constrained-policy-set* and put the result as the new value of *authority-constrained-policy-set*. Check that the intersection of *authority-constrained-policy-set* and *user-constrained-policy-set* is non-empty.

f) Check that the subject name located in the **subjectAltName** extension is within the name-space given by the value of **permitted-subtrees** and is not within the name-space given by the value of **excluded-subtrees**;

g) For an intermediate certificate, if the **keyUsage** extension is present and is flagged critical, verify that the **keyCertSign** bit is set.

If any of the above checks fail, the procedure terminates, returning a failure indication and an appropriate reason code. If the checks on the end certificate are successful, the procedure terminates, returning a success indication together with the set of policy identifiers from **authority-constrained-policy-set**, the associated policy element qualifiers and details of any policy mapping that may have occurred.

For an intermediate certificate, the following constraint recording actions are then performed, in order to correctly set up the state variables for the processing of the next certificate:

a) If the **nameConstraints** extension with a **permittedSubtrees** component is present in the certificate, set the **permitted-subtrees** state variable to the intersection of its previous value and the value indicated in the certificate extension.

b) If the **nameConstraints** extension with an **excludedSubtrees** component is present in the certificate, set the **excluded-subtrees** state variable to the union of its previous value and the value indicated in the certificate extension.

c) If the explicit-policy-indicator is not set:

   — if the *explicit-policy-pending* indicator is set, decrement the corresponding *skip-certificates* value and, if this value becomes zero, set *explicit-policy-indicator*.

   — If the **requireExplicitPolicy** constraint is present in the certificate, perform the following:

      i) For a **SkipCerts** value of 0, set *explicit-policy-indicator*.

      ii) For any other **SkipCerts** value, set the *explicit-policy-pending* indicator and set the corresponding *skip-certificates* value to the lesser of the **SkipCerts** value and the previous *skip-certificates* value (if the *explicit-policy-pending* indicator was already set).

d) If the policy-mapping-inhibit-indicator is not set:

   — process any policy mapping extension with respect to policies in the *user-constrained-policy-set* and add appropriate policy identifiers to the *user-constrained-policy-set*.

   — process any policy mapping extension with respect to policies in the *authority-constrained-policy-set* and add appropriate policy identifiers to the *authority-constrained-policy-set*.

   — if the *policy-mapping-inhibit-pending* indicator is set, decrement the corresponding *skip-certificates* value and, if this value becomes zero, set the *policy-mapping-inhibit-indicator*.

   — If the **inhibitPolicyMapping** constraint is present in the certificate, perform the following:

      i) For a **SkipCerts** value of 0, set the *policy-mapping-inhibit-indicator*.

      ii) For any other **SkipCerts** value, set the *policy-mapping-inhibit-pending* indicator and set the corresponding *skip-certificates* value to the lesser of the **SkipCerts** value and the previous *skip-certificates* value (if the *policy-mapping-inhibit-pending* indicator was already set).

9 Basic CRL extensions

9.1 Management requirements

The following requirements relate to CRLs:

a) When relying parties use CRLs, they must be able to detect a missing CRL. CRL sequence numbers are therefore required.

b) Some CRL users may wish to respond differently to a revocation, depending upon the reason for the revocation. There is therefore a requirement for a CRL entry to indicate the reason for revocation.

c) There may be a requirement for a CA to be able to temporarily suspend validity of a certificate and subsequently either revoke or reinstate it. Possible reasons for such an action include:

   — desire to reduce liability for erroneous revocation when a revocation request is unauthenticated and there is inadequate information to determine whether it is valid;

   — other business needs.

d) For each revoked certificate, a CRL contains the date when the CA posted the revocation.

e) Further information may be known as to when an actual or suspected key compromise occurred and this information may be valuable to a relying party. The revocation date is insufficient to solve some disputes because, assuming the worst, all signatures issued during the validity period of the certificate have to be considered invalid. However, it may be important for a user that a signed document be recognized as valid even though the key used to sign the message was compromised after the signature was produced. To assist in solving this problem, a CRL entry may include a second date (**invalidityDate**) which indicates when it was known or suspected that the private key was compromised.

9.2 Basic CRL and CRL entry extensions

9.2.1 Overview

The following extensions are defined:

— CRL number;

— reason code;

— hold instruction code;

— invalidity date.

The CRL number extension shall be used only as a CRL extension and the other fields shall be used only as CRL entry extensions.

9.2.2 CRL number extension

This CRL extension conveys a sequence number that increases by one for each CRL issued by a given CRL issuer through a given CA directory attribute or CRL distribution point. It allows a CRL user to detect whether CRLs issued prior to the one being processed were also seen and processed. This extension is defined as follows:

```
cRLNumber EXTENSION ::= {
    SYNTAX          CRLNumber
    IDENTIFIED BY   id-ce-cRLNumber }
```

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

This extension is always non-critical.

9.2.3 Reason code extension

This CRL entry extension identifies a reason for the certificate revocation. The reason code may be used by applications to decide, based on local policy, how to react to posted revocations. This extension is defined as follows:

```
reasonCode EXTENSION ::= {
    SYNTAX           CRLReason
    IDENTIFIED BY    id-ce-reasonCode }

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

When this extension is populated, one and only one of the following reason code values[6)] indicate why a certificate was revoked:

— **keyCompromise** is used in revoking an end entity certificate; it indicates that it is known or suspected that the subject's private key has been compromised;

— **cACompromise** is used in revoking a CA certificate; it indicates that it is known or suspected that the subject's private key has been compromised;

— **affiliationChanged** indicates that the subject is no longer affiliated with that CA (may be revoked by entity request);

— **superseded** indicates that the certificate has been superseded but there is no cause to suspect that the private key has been compromised;

— **cessationOfOperation** indicates that the entity, in the case of a natural person, is deceased and, in the case of a legal person, has ceased business operations;

— **certificateHold** indicates that the usage of the certificate is suspended.

  The certificate hold notice may include an optional hold instruction code to convey additional information to relying parties (see 9.2.4). Once a hold has been issued, it may be handled in one of three ways:

  a) it may remain on the CRL with no further action, causing users to reject transactions issued during the hold period; or

  b) it may be replaced by a (final) revocation for the same certificate, in which case the reason shall be one of the standard reasons for revocation, the revocation date shall be the date the certificate was placed on hold and the optional instruction code extension field shall not appear; or

  c) it may be explicitly released and the entry removed from the CRL;

— **removeFromCRL** indicates that an existing CRL entry should now be removed owing to certificate expiration or hold release. An entry with this reason code shall be used in delta-CRLs for which the corresponding base

---

6) ISO/IEC 9594-8 does not mandate the use of a single reason code, but implementations could be simplified by using only one of these extensions in each CRL entry.

CRL contains an entry for the same certificate with reason code **certificateHold**. This reason code is for use with delta-CRLs (see 10.1) only.

This extension is always non-critical.

9.2.4 Hold instruction code extension

This CRL entry extension provides for inclusion of a registered instruction identifier to indicate the action to be taken on encountering a held certificate. It is applicable only in an entry having a **certificateHold** reason code. This extension is defined as follows:

```
holdInstructionCode EXTENSION ::= {
      SYNTAX            HoldInstruction
      IDENTIFIED BY     id-ce-instructionCode }
```

```
HoldInstruction ::= OBJECT IDENTIFIER
```

This extension is always non-critical. Some hold instruction codes are defined in ISO/IEC 9594-8.

9.2.5 Invalidity date extension

This CRL entry extension indicates the date at which it is known or suspected that the private key was compromised or that the certificate should otherwise be considered invalid. This date may be earlier than the revocation date in the CRL entry, which is the date at which the CA processed the revocation. This extension is defined as follows:

```
invalidityDate EXTENSION ::= {
      SYNTAX            GeneralizedTime
      IDENTIFIED BY     id-ce-invalidityDate }
```

This extension is always non-critical.

When a revocation is first posted by a CA in a CRL, the invalidity date may precede the date of issue of earlier CRLs. The revocation date shall not precede the date of issue of earlier CRLs.

The date in this extension is not, by itself, sufficient for non-repudiation purposes. For example, this date may be a date advised by the private key holder and it is possible for such a person to fraudulently claim that a key was compromised some time in the past, in order to repudiate a validly-generated signature.

10 CRL distribution points and delta-CRLs

10.1 Requirements

Since revocation lists may become large and unwieldy, the ability to represent partial CRLs may be required.

The following requirements relate to CRL distribution points and delta-CRLs:

a) An option to control CRL sizes is to assign subsets of certificates issued by one CA to different CRLs. This can be achieved by associating every certificate with a CRL distribution point which may be, for example:

   — a directory entry whose CRL attribute will contain a revocation entry for that certificate, if it has been revoked; or

   — location such as an electronic mail address or Internet Uniform Resource Identifier from which the applicable CRL can be obtained.

b) For performance reasons, it may be desirable to reduce the number of CRLs that need to be checked when validating multiple certificates, e.g. a certification path. This can be achieved by having one CRL issuer sign and issue CRLs containing revocations from multiple CAs.

c) There may be a requirement for separate CRLs covering revoked CA certificates and revoked end entity certificates. This can facilitate processing of certification paths as the CRL for revoked CA certificates can be expected to be very short (usually empty). The **authorityRevocationList** and **certificateRevocationList** attributes have been specified for this purpose. However, for this separation to be secure, it is necessary to have an indicator in a CRL identifying which list it is. Otherwise, the substitution of one list for the other cannot be detected.

d) There may be a need for a different CRL to exist for potential compromise situations than which includes all routine revocations.

e) There may be a need for delta-CRLs which only contain entries for certificates that have been revoked and held and removed from the CRL since the issuance of a base CRL.

f) While requirements a) through e) may lead to optional revocation mechanisms, every CA shall, as a common fall-back approach, create complete CRLs. There is no requirement that CRLs covering all certificates that the CA issues be distributed.

10.2 Certificate extensions

The CRL distribution points extension shall be used only as a certificate extension and may be used in both CA certificates and end entity certificates. This extension identifies the CRL distribution point or points to which a relying party should refer to ascertain if the certificate has been revoked. A relying party can obtain a CRL from an applicable distribution point or it can obtain a current complete CRL from the CA directory entry. This extension is defined as follows:

```
cRLDistributionPoints EXTENSION ::= {
    SYNTAX          CRLDistPointsSyntax
    IDENTIFIED BY   id-ce-cRLDistributionPoints }

CRLDistPointsSyntax ::= 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) }
```

The **distributionPoint** component identifies the location from which the CRL can be obtained. If this component is absent, the distribution point name defaults to the CRL issuer name.

When the **fullName** alternative is used or when the default applies, the distribution point name may have multiple name forms. The same name, in at least one of its name forms, shall be present in the **distributionPoint** field of the issuing distribution point extension of the CRL. A certificate-using system is not required to be able to process all name forms. It may use a distribution point, provided at least one name form can be processed. If no name forms for a distribution point can be processed, a certificate-using system can still use the certificate, provided requisite revocation information can be obtained from another source, e.g. another distribution point or the CA's directory entry.

The **nameRelativeToCRLIssuer** component can be used only if the CRL distribution point is assigned a directory name that is directly subordinate to the directory name of the CRL issuer. In this case, the **nameRelativeToCRLIssuer** component conveys the relative distinguished name with respect to the CRL issuer directory name.

The **reasons** component indicates the revocation reasons covered by this CRL. If the **reasons** component is absent, the corresponding CRL distribution point distributes a CRL which will contain an entry for this certificate if this certificate has been revoked, regardless of revocation reason. Otherwise, the **reasons** value indicates which revocation reasons are covered by the corresponding CRL distribution point. Only one of the **ReasonFlags** shall be set.7)

The **cRLIssuer** component identifies the authority that issues and signs the CRL. If this component is absent, the CRL issuer name defaults to the certificate issuer name.

This extension may, at the option of the certificate issuer, be either critical or non-critical. In the interests of interoperability, it is recommended that it be flagged non-critical. If the CRL distribution point extension is made critical, the CA shall ensure that the distribution points include CRL entries for reason codes **keyCompromise** and/or **cACompromise,** whichever is applicable.

If this extension is flagged critical, then a certificate-using system shall not use the certificate without first retrieving and checking a CRL from one of the nominated distribution points whose CRL entries include, at a minimum, those with reason codes **keyCompromise** (for an end entity certificate) or **cACompromise** (for a CA certificate).

If this extension is flagged non-critical and a certificate-using system does not recognize the extension type, then that system should only use the certificate if:

— it can check a complete CRL from the CA;

— revocation checking is not required under local policy; or

— revocation checking is accomplished by other means.

NOTE It is possible to have CRLs issued by more than one CRL issuer for the one certificate. Co-ordination of these CRL issuers and the issuing CA is an aspect of CA policy.

10.3 CRL and CRL entry extensions

10.3.1 Overview

The following CRL or CRL entry extensions are defined:

— issuing distribution point;

— certificate issuer;

— delta CRL indicator.

Issuing distribution point and delta CRL indicator shall be used only as CRL extensions. Certificate issuer shall be used only as a CRL entry extension.

7) They are mutually exclusive.

10.3.2 Issuing distribution point extension

This CRL extension identifies the CRL distribution point for this particular CRL and indicates if the CRL is limited to revocations for end entity certificates only, for CA certificates only, or for a limited set of reasons only. The CRL is signed by the CRL issuer's key — CRL distribution points do not have their own key pairs. However, for a CRL distributed via an X.500 Directory, the CRL is stored in the entry of the CRL distribution point, which may not be the directory entry of the CRL issuer.

This extension is defined as follows:

```
issuingDistributionPoint EXTENSION ::= {
    SYNTAX          IssuingDistPointSyntax
    IDENTIFIED BY   id-ce-issuingDistributionPoint }

IssuingDistPointSyntax ::= SEQUENCE {
    distributionPoint         [0] DistributionPointName OPTIONAL,
    onlyContainsUserCerts     [1] BOOLEAN DEFAULT FALSE,
    onlyContainsCACerts       [2] BOOLEAN DEFAULT FALSE,
    onlySomeReasons           [3] ReasonFlags OPTIONAL,
    indirectCRL               [4] BOOLEAN DEFAULT FALSE }
```

The **distributionPoint** component contains the name of the distribution point in one or more name forms. If this extension is absent, the CRL shall contain entries for all revoked unexpired certificates issued by the CRL issuer.

If **onlyContainsUserCerts** is true, the CRL only contains revocations for end entity certificates. If **onlyContainsCACerts** is true, the CRL only contains revocations for CA certificates. **onlyContainsUserCerts** and **onlyContainsCACerts** shall not both be set[8]. If **onlySomeReasons** is present, the CRL only contains revocations for the identified reason or reasons, otherwise the CRL contains revocations for all reasons.

If **indirectCRL** is true, then the CRL may contain revocation notifications from CAs other than the issuer of the CRL. The particular CA responsible for each entry is as indicated by the certificate issuer CRL entry extension in that entry or in accordance with the defaulting rules described in 10.3.3. In such a CRL, it is the responsibility of the CRL issuer to ensure that the CRL is complete in that it contains all revocation entries, consistent with **onlyContainsUserCerts**, **onlyContainsCACerts** and **onlySomeReasons** indicators, from all CAs that identify this CRL issuer in their certificates.

For CRLs distributed via an X.500 Directory, the following rules regarding use of attributes apply. A CRL which has **onlyContainsCACerts** set shall be distributed via the **authorityRevocationList** attribute of the associated distribution point or, if no distribution point is identified, via the **authorityRevocationList** attribute of the CRL issuer entry. Otherwise, the CRL shall be distributed via the **certificateRevocationList** attribute of the associated distribution point or, if no distribution point is identified, via the **certificateRevocationList** attribute of the CA entry.

This extension is always critical. A relying party which does not implement this extension cannot assume that the CRL contains a complete list of revoked certificates of the identified CA. CRLs not containing critical extensions shall contain all current CRL entries for the issuing CA, including entries for all revoked user certificates and CA certificates.

10.3.3 Certificate issuer extension

This CRL entry extension identifies the certificate issuer associated with an entry in an indirect CRL, i.e. a CRL that has the **indirectCRL** indicator set in its issuing distribution point extension. If this extension is not present on the first entry in an indirect CRL, the certificate issuer defaults to the CRL issuer. On subsequent entries in an indirect CRL, if this extension is not present, the certificate issuer for the entry is the same as that for the preceding entry. This extension is defined as follows:

8) They are mutually exclusive.

```
certificateIssuer EXTENSION ::= {
    SYNTAX          GeneralNames
    IDENTIFIED BY   id-ce-certificateIssuer }
```

This extension is always critical. If an implementation ignored this extension, it could not correctly attribute CRL entries to certificates.

10.3.4 Delta CRL indicator extension

This CRL extension identifies a CRL as being a delta-CRL only. This extension is defined as follows:

```
deltaCRLIndicator EXTENSION ::= {
    SYNTAX          BaseCRLNumber
    IDENTIFIED BY   id-ce-deltaCRLIndicator }

BaseCRLNumber ::= CRLNumber
```

The value of type **BaseCRLNumber** identifies the CRL number of the base CRL that was used as the starting point in the generation of this delta-CRL, i.e. this delta-CRL contains the changes between the base CRL and the complete CRL issued along with this delta-CRL. It is the decision of a CA as to whether to provide delta-CRLs. However, a delta-CRL shall not be issued without a corresponding complete CRL being issued at the same time. The value of the CRL number, as conveyed in the CRL number extension (if present), shall be identical for both the delta-CRL and the corresponding complete CRL.

The CRL user constructing a locally held CRL from delta-CRLs shall consider the constructed CRL incomplete and unusable if both the following conditions are true:

— the value of the **CRLNumber** of the received delta-CRL is more than one greater than that of the **CRLNumber** of the delta-CRL last processed;

— the value of **BaseCRLNumber** of the received delta-CRL has changed from the **BaseCRLNumber** of the delta-CRL last processed.

This extension is always critical. A relying party that does not understand the use of delta-CRLs should not use a CRL containing this extension, as the CRL may not be as complete as the user expects. CRLs not containing critical extensions shall contain all current CRL entries for the issuing CA, including entries for all revoked user certificates and CA certificates.

NOTE It is the decision of the CA as to whether or not to distribute the CRLs issued between two base CRLs. For example, it may be the policy of the CA to distribute base CRLs via CD-ROM and delta-CRLs via an on-line service such as the Directory. Although the CA had issued its CRL with the associated delta-CRL, it may be the CA's policy that the user shall construct the current CRL by applying the delta-CRL to the base CRL held on the CD-ROM.

10.4 Matching rules

10.4.1 Overview

This subclause defines matching rules for use with attributes of type:

— **Certificate**;

— **CertificatePair**;

— **CertificateList**;

— **SupportedAlgorithm**.

10.4.2 Certificate exact match

The certificate exact match rule compares for equality a presented value with an attribute value of type **Certificate**. It uniquely selects a single certificate.

```
certificateExactMatch MATCHING-RULE ::= {
    SYNTAX      CertificateExactAssertion
    ID          id-mr-certificateExactMatch }

CertificateExactAssertion ::= SEQUENCE {
    serialNumber          CertificateSerialNumber,
    issuer                Name }
```

This matching rule returns **TRUE** if the components in the attribute value match those in the presented value.

10.4.3 Certificate match

The certificate match rule compares a presented value with an attribute value of type **Certificate**. It selects one or more certificates on the basis of various characteristics.

```
certificateMatch MATCHING-RULE ::= {
    SYNTAX      CertificateAssertion
    ID          id-mr-certificateMatch }

CertificateAssertion ::= SEQUENCE {
    serialNumber              [0] CertificateSerialNumber    OPTIONAL,
    issuer                    [1] Name      OPTIONAL,
    subjectKeyIdentifier      [2] SubjectKeyIdentifier   OPTIONAL,
    authorityKeyIdentifier    [3] AuthorityKeyIdentifier      OPTIONAL,
    certificateValid          [4] Time       OPTIONAL,
    privateKeyValid           [5] GeneralizedTime        OPTIONAL,
    subjectPublicKeyAlgID     [6] OBJECT IDENTIFIER  OPTIONAL,
    keyUsage                  [7] KeyUsage      OPTIONAL,
    subjectAltName            [8] AltNameType       OPTIONAL,
    policy                    [9] CertPolicySet            OPTIONAL,
    pathToName                [10] Name    OPTIONAL }

AltNameType ::= CHOICE {
    builtinNameForm      ENUMERATED {
        rfc822Name                    (1),
        dNSName                       (2),
        x400Address                   (3),
        directoryName                 (4),
        ediPartyName                  (5),
        uniformResourceIdentifier     (6),
        iPAddress                     (7),
        registeredId                  (8) },
    otherNameForm        OBJECT IDENTIFIER }

CertPolicySet ::= SEQUENCE (1..MAX) OF CertPolicyId
```

This matching rule returns **TRUE** if all of the components that are present in the presented value match the corresponding components of the attribute value, in accordance with the following rules:

a) **serialNumber** matches if the value of this component in the attribute value equals that in the presented value;

b) **issuer** matches if the value of this component in the attribute value equals that in the presented value;

c) **subjectKeyIdentifier** matches if the value of this component in the stored attribute value equals that in the presented value; there is no match if the stored attribute value contains no subject key identifier extension;

d) **authorityKeyIdentifier** matches if the value of this component in the stored attribute value equals that in the presented value; there is no match if the stored attribute value contains no authority key identifier extension or if not all components in the presented value are present in the stored attribute value;

e) **certificateValid** matches if the presented value falls within the validity period of the stored attribute value;

f) **privateKeyValid** matches if the presented value falls within the period indicated by the private key usage period extension of the stored attribute value or if there is no private key usage period extension in the stored attribute value;

g) **subjectPublicKeyAlgID** matches if it is equal to the **algorithm** component of the **algorithmIdentifier** of the **subjectPublicKeyInformation** component of the stored attribute value;

h) **keyUsage** matches if all of the bits set in the presented value are also set in the key usage extension in the stored attribute value, or if there is no key usage extension in the stored attribute value;

i) **subjectAltName** matches if the stored attribute value contains the subject alternative name extension with an **AltNames** component of the same name type as indicated in the presented value;

j) **policy** matches if all of the policy elements identified in one of the presented values are contained in the set of **policyElementIds** in any of the **policyInformation** values in the certificate policies extension in the stored attribute value; there is no match if there is no certificate policies extension in the stored attribute value;

k) **pathToName** matches unless the certificate has a name constraints extension which inhibits the construction of a certification path to the presented name value.

10.4.4 Certificate pair exact match

The certificate pair exact match rule compares for equality a presented value with an attribute value of type **CertificatePair**. It uniquely selects a single cross-certificate pair.

```
certificatePairExactMatch MATCHING-RULE ::=     {
    SYNTAX          CertificatePairExactAssertion
    ID          id-mr-certificatePairExactMatch }

CertificatePairExactAssertion ::= SEQUENCE {
    forwardAssertion            [0] CertificateExactAssertion OPTIONAL,
    reverseAssertion            [1] CertificateExactAssertion OPTIONAL }
    (WITH COMPONENTS            {..., forwardAssertion PRESENT} |
    WITH COMPONENTS             {..., reverseAssertion PRESENT} )
```

This matching rule returns **TRUE** if the components that are present in the **forwardAssertion** and **reverseAssertion** components of the presented value match the corresponding components of the **forward** and **reverse** components, respectively, in the stored attribute value.

10.4.5 Certificate pair match

The certificate pair match rule compares a presented value with an attribute value of type **CertificatePair**. It selects one or more cross-certificate pairs on the basis of various characteristics of either the forward or reverse certificate of the pair.

```
certificatePairMatch MATCHING-RULE ::= {
    SYNTAX      CertificatePairAssertion
    ID          id-mr-certificatePairMatch }
```

```
CertificatePairAssertion ::= SEQUENCE {
      forwardAssertion          [0] CertificateAssertion OPTIONAL,
      reverseAssertion          [1] CertificateAssertion OPTIONAL }
      ( WITH COMPONENTS    {..., forwardAssertion PRESENT} |
      WITH COMPONENTS            {..., reverseAssertion PRESENT} )
```

This matching rule returns **TRUE** if all of the components that are present in the **forwardAssertion** and **reverseAssertion** components of the presented value match the corresponding components of the **forward** and **reverse** components, respectively, in the stored attribute value, using the rules given in 10.4.3.

10.4.6 Certificate list exact match

The certificate list exact match rule compares for equality a presented value with an attribute value of type **CertificateList**. It uniquely selects a single CRL.

```
certificateListExactMatch MATCHING-RULE ::=      {
      SYNTAX      CertificateListExactAssertion
      ID          id-mr-certificateListExactMatch }

CertificateListExactAssertion ::= SEQUENCE {
      issuer                   Name,
      thisUpdate               Time,
      distributionPoint        DistributionPointName OPTIONAL }
```

The rule returns **TRUE** if the components in the stored attribute value match those in the presented value. If the **distributionPoint** component is present, then it shall match in at least one name form.

10.4.7 Certificate list match

The certificate list match rule compares a presented value with an attribute value of type **CertificateList**. It selects one or more CRLs on the basis of various characteristics.

```
certificateListMatch MATCHING-RULE ::= {
      SYNTAX      CertificateListAssertion
      ID          id-mr-certificateListMatch }

CertificateListAssertion ::= SEQUENCE {
      issuer                   Name    OPTIONAL,
      minCRLNumber             [0]   CRLNumber OPTIONAL,
      maxCRLNumber             [1]   CRLNumber OPTIONAL,
      reasonFlags              ReasonFlags      OPTIONAL,
      dateAndTime              Time      OPTIONAL,
      distributionPoint        [2]   DistributionPointName OPTIONAL }
```

The matching rule returns **TRUE** if all of the components that are present in the presented value match the corresponding components of the stored attribute value, as follows:

a) **issuer** matches if the value of this component in the attribute value equals that in the presented value;

b) **minCRLNumber** matches if its value is less than or equal to the value in the CRL number extension of the stored attribute value; there is no match if the stored attribute value contains no CRL number extension;

c) **maxCRLNumber** matches if its value is greater than or equal to the value in the CRL number extension of the stored attribute value; there is no match if the stored attribute value contains no CRL number extension;

d) **reasonFlags** matches if any of the bits that are set in the presented value are also set in the **onlySomeReasons** components of the issuing distribution point extension of the stored attribute value; there is no match if the stored attribute value contains no issuing distribution point extension;

e) **dateAndTime** matches if the value is equal to or later than the value in the **thisUpdate** component of the stored attribute value and is earlier than the value in the **nextUpdate** component of the stored attribute value; there is no match if the stored attribute value contains no **nextUpdate** component;

f) **distributionPoint** matches if the stored attribute value contains an issuing distribution point extension and the value of this component in the presented value equals the corresponding value, in at least one name form, in that extension.

10.4.8 Algorithm identifier match

The algorithm identifier match rule compares for equality a presented value with an attribute value of type **SupportedAlgorithm**.

```
algorithmIdentifierMatch MATCHING-RULE ::=  {
    SYNTAX      AlgorithmIdentifier
    ID          id-mr-algorithmIdentifierMatch }
```

The rule returns TRUE if the presented value is equal to the **algorithmIdentifier** component of the stored attribute value.

Annex A
(informative)

Examples of the use of certification path constraints

A.1 Example 1: Use of basic constraints

Suppose the Widget Corporation wants to cross-certify the central CA of the Acme Corporate Group, but only wants the Widget community to use end entity certificates issued by that CA, not certificates issued by other CAs certified by that CA.

The Widget Corporation could satisfy this requirement by issuing a certificate for Acme's central CA, including the following extension field value:

Value of Basic Constraints field:

{ cA TRUE, pathLenConstraint 0 }

A.2 Example 2: Use of name constraints

Suppose the Widget Corporation wants to cross-certify the central CA of the Acme Corporate Group, but only wants the Widget community to use Acme certificates for subjects that meet the following criteria:

— in Acme, Inc. in the U.S., all subjects are acceptable except for subjects in purchasing;

— in EuroAcme in France, only those subjects that are immediately subordinate to the EuroAcme headquarters are acceptable (this includes individuals reporting directly to headquarters but excludes those reporting to subordinate organizations);

— in Acme Ltd. in the U.K., all subjects are acceptable except those reporting to organizations that are subordinate to the R&D organizational unit (this includes individuals reporting directly to R&D but excludes those reporting to subordinate units of R&D).

The Widget Corporation could satisfy these requirements by issuing a certificate for Acme's central CA, including the following extension field values:

Value of Basic Constraints field:

{ cA TRUE }

Value of Name Constraints field:

```
{ permittedSubtrees {{base --Country=US, Org=Acme Inc--},
    {base --Country=France, Org=EuroAcme--, maximum 1},
    {base --Country=UK, Org=Acme Ltd--}},
    excludedSubtrees {{base --Country=US, Org=Acme Inc, Org. Unit=Purchasing-},
    {base --Country=UK Org=Acme Ltd., Org. Unit=R&D--, minimum 2}}}
```

A.3 Example 3: Use of policy mapping and policy constraints

Suppose the following cross-certification scenario is required between the Canadian and U.S. governments:

1) a Canadian government CA wishes to certify use of U.S. government signatures with respect to a Canadian policy called *Can/US-Trade*;

2) the U.S. government has a policy called *US/Can-Trade*, which the Canadian government is prepared to consider equivalent to its *Can/US-Trade* policy;

3) the Canadian government wants to apply safeguards which require all U.S. certificates to explicitly state support for the policy and which inhibit mapping to other policies within the U.S. domain.

A Canadian government CA could issue a certificate for a U.S. government CA with the following extension field values:

Value of Certificate Policies field:

{{ policyIdentifier -- object identifier for Can/US-Trade -- }}

Value of Policy Mappings field:

**{{ issuerDomainPolicy -- object identifier for Can/US-Trade -- ,
subjectDomainPolicy -- object identifier for US/Can-Trade -- }}**

Value of **PolicyConstraints** field:

**{{ policySet { -- object identifier for Can/US-Trade -- }, requireExplicitPolicy (0),
inhibitPolicyMapping (0)}}**

Bibliography

[1] ISO/IEC 8824-1:1998 | ITU-T Recommendation X.680 (1997), *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation*

[2] ISO/IEC 8824-2:1998 | ITU-T Recommendation X.681 (1997), *Information technology — Abstract Syntax Notation One (ASN.1): Information object specification*

[3] ISO/IEC 8824-3:1998 | ITU-T Recommendation X.682 (1997), *Information technology — Abstract Syntax Notation One (ASN.1): Constraint specification*

[4] ISO/IEC 8824-4:1998 | ITU-T Recommendation X.683 (1997), *Information technology — Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications*

[5] ISO/IEC 8825-1:1998 | ITU-T Recommendation X.690 (1997), *Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

[6] ISO/IEC 8825-2:1998 | ITU-T Recommendation X.691 (1997), *Information technology — ASN.1 encoding rules: Specification of Packed Encoding Rules (PER)*

[7] ISO/IEC 9834-1, *Information technology — Open Systems Interconnection — Procedures for the operation of OSI Registration Authorities: General procedures*

[8] ANSI X9 TG-9-1995, *Abstract Syntax Notation and Encoding Rules for Inf. Ind. Standards*

[9] ISO/IEC 10118-3:1998, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*

ICS 35.240.40

Price based on 38 pages