INTERNATIONAL STANDARD

ISO 15782-1

First edition
2003-06-01

Certificate management for financial services —

Part 1: Public key certificates

Gestion de certificats pour les services financiers —

Partie 1: Certificats de clé publique

Reference number
ISO 15782-1:2003(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

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 2.

The main task of technical committees is to prepare International Standards. 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 document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 15782-1 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 *Certificate management for financial services*:

— *Part 1: Public key certificates*

— *Part 2: Certificate extensions*

Introduction

This part of ISO 15782 adopts ISO/IEC 9594-8 | ITU-T Recommendation X.509 for the financial services industry and defines certificate management procedures and data elements.

Detailed requirements for the financial industry for the individual extensions are given in ISO 15782-2.

While the techniques specified in this part of ISO 15782 are designed to maintain the integrity of financial messages and support the service of non-repudiation, this part of ISO 15782 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, with these controls including 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 is documented in order to prove the ownership of the corresponding private key. This binding is called a *public key certificate*. Public key certificates are generated by a trusted entity known as a Certification Authority (CA).

The proper implementation of this part of ISO 15782 ought to provide assurances of the binding of the identity of an entity to the key used by that entity to sign documents, including wire transfers and contracts.

This part of ISO 15782 defines a certificate management framework for authentication, including the authentication of keys for encryption. The techniques specified by this part of ISO 15782 can be used when initiating a business relationship between legal entities (entities).

Certificate management for financial services —

Part 1:
Public key certificates

1 Scope

This part of ISO 15782 defines a certificate management system for financial industry use for legal and natural persons that includes

— credentials and certificate contents,

— certification authority systems, including certificates for digital signatures and for encryption key management,

— certificate generation, distribution, validation and renewal,

— authentication structure and certification paths,

— revocation and recovery procedures, and

— extensions to the definitions of public key certificates and certificate revocation lists.

This part of ISO 15782 also recommends some useful operational procedures (e.g. distribution mechanisms, acceptance criteria for submitted credentials).

Implementation of this part of ISO 15782 will also be based on business risks and legal requirements.

This part of ISO 15782 does not include

— the protocol messages used between the participants in the certificate management process,

— requirements for notary and time stamping,

— certificate policy and certification practices requirements,

— requirements for trusted third parties, or

— Attribute Certificates.

While this part of ISO 15782 provides for the generation of certificates that could include a public key used for encryption key management, it does not address the generation or transport of keys used for encryption.

Implementers wishing to comply with ISO/IEC 9594-8 | ITU-T Recommendation X.509 can utilize the certificate structures defined by that International Standard. Those wishing to implement compatible certificate and certificate revocation structures but without the overhead associated with the X.500 series can utilize the ASN.1 structures defined in Annex A.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

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

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

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

ISO/IEC 8825-1 | 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) — Part 1*

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

ISO/IEC 9594-2 | ITU-T Recommendation X.501 (1997), *Information technology — Open Systems Interconnection — The Directory: Models — Part 2*

ISO/IEC 9594-6 | ITU-T Recommendation X.520 (1997), *Information technology — Open Systems Interconnection — The Directory: Selected attribute types — Part 6*

ISO/IEC 9594-8 | ITU-T Recommendation X.509 (1997), *Information Technology — Open Systems Interconnection — The Directory: Public-key and attribute certificate frameworks — Part 8*

ISO 9807:1991, *Banking and related financial services — Requirements for message authentication (retail)*

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

ISO/IEC 15408 (all parts), *Common Criteria for Information Security Evaluation*

ISO 15782-2:2001, *Banking — Certificate Management — Part 2: Certificate extensions*

ANS X9.30-1, *Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry, Part 1: The Digital Signature Algorithm (DSA)*

ANS X9.31-1, *Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry, Part 1: The RSA Signature Algorithm*

ANS X9.62, *Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
ASN.1 module
identifiable collection of ASN.1 types and values

3.2
attribute
characteristic of an entity

3.3
audit journal
chronological record of system activities which is sufficient to enable the reconstruction, review and examination of the sequence of environments and activities surrounding or leading to each event in the path of a transaction from its inception to the output of the final results

3.4
authorization
granting of rights

3.5
CA-certificate
certificate whose subject is a CA, and whose associated private key is used to sign certificates

3.6
(certificate) hold
suspension of the validity of a certificate

3.7
certificate information
information in a certificate which is signed

3.8
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. Some of the components in the certificate policy framework are given concrete values and 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).

3.9
certificate policy framework
comprehensive set of security and liability related components that can be used to define a certificate policy

NOTE A subset of the components in the certificate policy framework are given concrete values to define a certificate policy.

3.10
certificate request data
credentials
signed information in a certificate request, including the entity's public key, entity identity and other information included in the certificate

3.11
certificate revocation list
CRL
list of revoked certificates

3.12
certificate-using system
implementation of those functions defined in this part of ISO 15782 that are used by a certificate user

3.13
certification
process of creating a public key certificate for an entity

3.14
certification authority
CA
entity trusted by one or more entities to create, assign, and revoke or hold public key certificates

3.15
certification authority system
set of entities, including a CA, that manages certificates throughout the life of the certificate

NOTE The entities are responsible for

— generation,

— submission,

— registration,

— certification,

— distribution,

— use,

— renewal,

— revocation or hold, and

— expiry.

3.16
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

3.17
certification practice statement
CPS
statement of the practices which a certification authority employs in issuing certificates

3.18
compromise
violation of the security of a system such that an unauthorized disclosure of sensitive information may have occurred

3.19
confidentiality
property that information is not made available or disclosed to unauthorized individuals, entities, or processes

3.20
CRL distribution point
directory entry or other distribution source for 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.

3.21
cross certification
process by which two CAs mutually certify each other's public keys

cf. **policy mapping** (3.48)

3.22
(cryptographic) key
parameter that determines the operation of a cryptographic function

NOTE Cryptographic functions include the following:

— the transformation from plain text to cipher text and vice versa;

— synchronized generation of keying material;

— digital signature generation or validation.

3.23
cryptographic module
device wherein cryptographic functions (e.g. encryption, authentication, key generation) are performed

3.24
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

3.25
cryptoperiod
time span during which a specific key is authorized for use or in which the keys for a given system may remain in effect

3.26
data integrity
property whereby data has not been altered or destroyed

3.27
delta-CRL
partial CRL indicating only changes since a prior CRL issue

3.28
(digital) 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

3.29
directory
repository
method for distributing or making available certificates or CRLs

EXAMPLE A data base or an X.500 Directory.

3.30
distinguished name
globally unique name for an entity

NOTE 1 Methods for determining global uniqueness are outside the scope of this part of ISO 15782.

NOTE 2 An entity may be issued more than one certificate with the same distinguished name.

3.31
dual control
process of utilizing two or more separate entities (usually persons), who are operating in concert, to protect sensitive functions or information

NOTE 1 Both entities are equally responsible for the physical protection of materials involved in vulnerable transactions. No single person is able to access or to utilize the materials (e.g. cryptographic key).

NOTE 2 For manual key and certificate generation, conveyance, loading, storage, and retrieval, dual control requires split knowledge of key among the entities. Also see split knowledge.

3.32
end certificate
final certificate considered in a certificate chain

3.33
end entity
certificate subject, other than a CA, which uses its private key for purposes other than signing certificates

3.34
entity
legal (e.g. a corporation, labour union, state or nation) or natural person

EXAMPLE CA, RA or end entity.

3.35
financial message
communication containing information which has financial implications

3.36
hash
(mathematical) function which maps values from a large (possibly very large) domain into a smaller (fixed) range and satisfies the following properties:

— it is computationally infeasible to find any input which maps to a pre-specified output;

— it is computationally infeasible to find any two distinct inputs which map to the same output

3.37
key agreement
method for negotiating a key value on-line without transferring the key, even in an encrypted form

EXAMPLE The Diffie-Hellman technique.

3.38
key fragment
part of a private key which has been divided into pieces (also called shares) that are distributed amongst entities such that the pooled fragments of specific subsets of entities can generate digital signatures of the original private key

3.39
key management
generation, storage, secure distribution, and application of keying material in accordance with a security policy

3.40
key pair
⟨public key cryptography⟩ public key and its corresponding private key

3.41
keying material
data, such as keys, certificates and initialization vectors, necessary to establish and maintain cryptographic keying relationships

3.42
keying relationship
state existing between a communicating pair or between members of a group (logical entity) during which time they share keying material

3.43
message
data to be signed

3.44
non-repudiation
service which provides proof of the integrity and origin of data which can be verified by a third party

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

3.45
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”.

3.46
out-of-band notification
notification using a communication means independent of the primary communications means

3.47
policy mapping
recognition that, when a 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

cf. **cross certification** (3.21)

3.48
policy qualifier
policy-dependent information that accompanies a certificate policy identifier in an X.509 certificate

3.49
private key
⟨asymmetric (public) key cryptosystem⟩ key of an entity’s key pair which is known only by that entity

3.50
public key
⟨asymmetric (public) key cryptosystem⟩ key of an entity’s key pair which is publicly known

3.51
public key 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

3.52
public key validation
PKV
process that does arithmetic tests on a candidate public key to provide assurance that it conforms to the specifications of the standard

NOTE 1 Since attacks may be possible on the owner and/or user if using a non-conforming public key.

NOTE 2 Public key validation can include arithmetic property tests (range, order, primality, etc.), canonical generation tests and consistency tests between components of a public key. Methods for public key validation are typically found in financial industry signature standards (e.g. ANS X9.62).

3.53
registration authority
RA
entity that is responsible for identification and authentication of subjects of certificates, but is not a CA and hence does not sign or issue certificates

NOTE An RA may assist in the certificate application process, revocation process or both.

3.54
relying party
user
recipient of a certificate who acts in reliance on that certificate

3.55
split knowledge
condition under which two or more entities separately have key fragments which, individually, convey no knowledge of the resultant cryptographic key

3.56
subject
entity whose public key is certified in a public key certificate

3.57
subject CA
CA that is certified by the issuing CA

3.58
trusted CA public key
public key used to validate the first certificate in a chain of certificates as a part of certification path processing

EXAMPLE The root key in a centralized trust model or a local CA key in a decentralized trust model (see Annex G).

NOTE If an end entity validates a chain of certificates from a trusted CA public key to the end certificate, then the end certificate is considered valid.

3.59
zeroize
active destruction of electronically stored data such as by degaussing, erasing or overwriting

4 Abbreviations and symbols

Abbreviation	Meaning
ASN.1	Abstract syntax notation
BER	Basic encoding rules
CA	Certification authority
CPS	Certification practice statement
CRL	Certificate revocation list
DER	Distinguished encoding rules
DSA	Digital signature algorithm
ECDSA	Elliptic curve digital signature algorithm
ITU-T	International Telecommunication Union telecommunications standardization sector
PKI	Public key infrastructure
RA	Registration authority
RSA	Rivest Shamir Adleman algorithm
SHA-1	Secure hash algorithm-1
URI	Uniform resource identifier

Symbols	Meaning
X{information}	Signing of “information” by X
X_p	X’s public key. (e.g. X_{1p} is X_1’s public key)
X_s	X’s private key
$X_1«X_2»$	X_2’s certificate issued by the CA, X_1
$X_1«X_2»X_2«X_3»X_{n-1}«X_n»$	Certificate path. Each item in the path is the certificate for the CA which produced the next item. This path is of arbitrary length and is functionally equivalent to $X_1«X_n»$. Possession of X_{1p} allows a user to extract the authenticated public key of X_n.
$X_{1p}·X_1«X_2»$	Unwrapping of a certificate or path. The public key of the leftmost CA (X_1) is used to extract the authenticated public key of the rightmost certificate ($X_1«X_2»$) by working through the path of intervening certificates. This example extracts X_{2p}.

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), as defined in ISO/IEC 8824-1 to ISO/IEC 8824-4 and ISO/IEC 8825-1 and ISO/IEC 8825-2. Where it makes sense to do so, the ASN.1 term is used in place of normal text.

5 Public key infrastructure

5.1 Overview

Public key infrastructure (PKI) is a term used to describe the technical, legal and commercial infrastructure that enables the wide deployment of public key technology.

Public key technology is used for creating digital signatures and for managing symmetric keys. With public key cryptography, two keys are used: one is kept private with the user, the other is made publicly available. That

which is signed or processed with one key (public or private) may be validated with its complement (public or private). Revealing the public key does not in any way compromise the private key.

The authentication of public keys is an essential requirement and for this reason public keys are housed in public key certificates. A certificate contains the public key and its identifying data and is digitally signed by a certification authority (CA). This part of ISO 15782 is based on the X.509 (1997) format for public key certificates.

5.2 Public key management infrastructure process flow

The responsibilities, services and procedures required by a public key management infrastructure are as follows:

— key generation;

— registration;

— certification;

— distribution;

— usage;

— revocation hold;

— expiry;

— renewal.

The main steps involved in certification are shown in Figure 1.

5.3 Certification Authority (CA)

A CA has a public/private key pair and uses a digital signature algorithm to produce certificates.

The binding of the entity's public key to its identity is accomplished by having the CA generate the certificate, thereby attesting to the relationship of the information therein and providing assurances of its integrity.

The binding of an entity's public key and identity is validated by using the public key of one or more CAs as described in 6.3.1. The certificate(s) and proof of validation shall be maintained by the validator in an audit journal. A CA may issue certificates to any entities, including CAs.

Entities (including CAs) can use these certificates to authenticate themselves to relying parties. Hence, authentication may involve a chain of certificates. The verification of a chain of certificates begins with the trusted CA public key and ends with the certificate being validated. The trusted CA public key shall be obtained and authenticated by some means other than by the use of certificates. This is to ensure that the process begins securely. See 6.3.1, Annex H and ISO/IEC 9594-8.

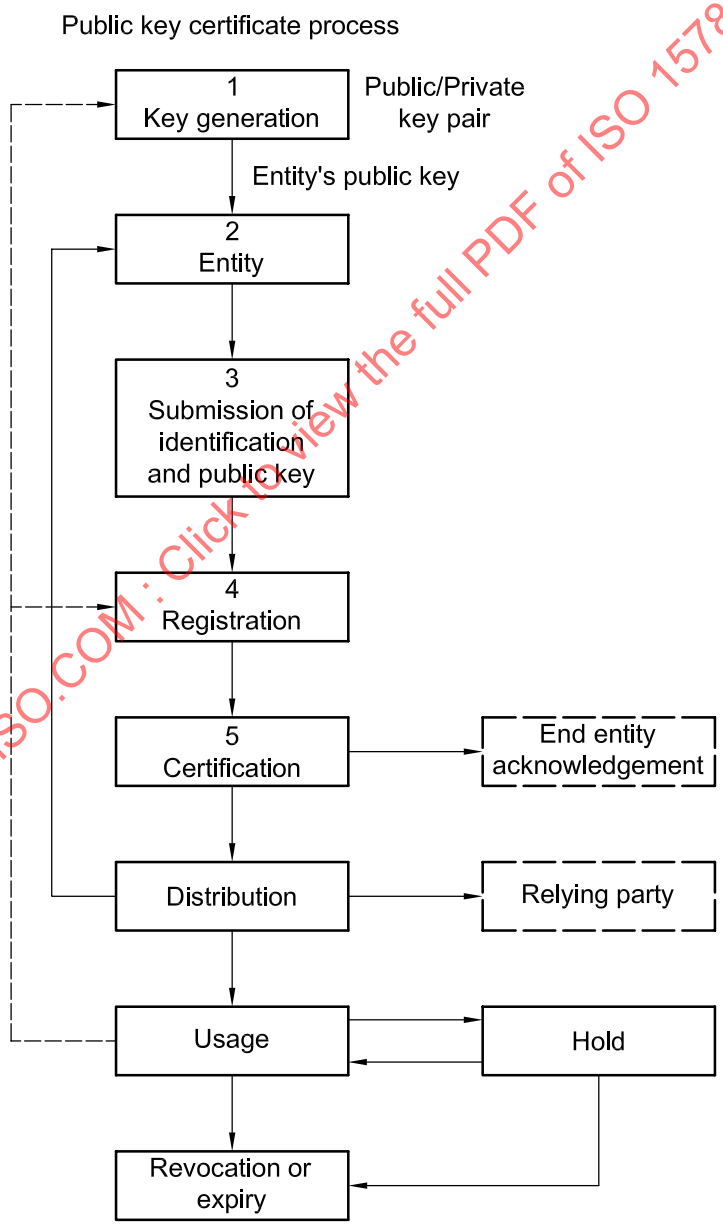
Once a certificate has been generated, the integrity of its contents is protected. This part of ISO 15782 does not require that certificates be given confidentiality protection. A valid copy of the CA's public key is required by the relying party in order to validate a certificate. Given that the CA is a trusted entity, this permits the validation of the binding between an entity's public key, its identity, and other needed information.

Two general architectures may be configured for certification paths: hierarchical and non-hierarchical. In a hierarchical architecture, authorities are arranged under a "root" CA that issues certificates to subordinate CAs. These subordinate CAs may issue certificates to CAs subordinate to them or to end entities. In a hierarchical architecture the public key of the root CA functions as the trusted CA public key and is known to every entity. Any entity's certificate may be validated by validating the certification path of signature certificates that leads from the certificate being validated back to the trusted CA public key of the root CA. In this architecture, the root CA is a mutual point of trust for all entities.

To communicate outside the root CA's domain, the root CA shall cross-certify with the desired remote domain. Certification path validation then involves building a chain of certificates from the remote entity to the root CA by way of the cross-certified remote CA.

In a non-hierarchical architecture, independent CAs may cross-certify each other by issuing public key certificates to each other. This results in a general network of trust relationships between CAs and allows each group (such as a retail credit authorization network, a clearing house, a financial institution or a subgroup thereof) to have its own CA. An entity uses the public key of a selected CA for its trusted CA public key. The certification path consists of those certificates that chain back from the certificate being validated to the trusted CA of the relying party.

A compromise of the private key of a CA compromises all users of certificates of the CA, because the holder of that private key can generate fraudulent certificates and then masquerade as one or more end entities. Failure to provide compensating controls to deal with the possibility of compromise of transactions in a financial network can have catastrophic effects on financial institutions and their customers.



NOTE Numbers correspond to process steps detailed in Figures 2 and 3.

Figure 1 — Typical public key management infrastructure process flow

5.4 Registration Authority (RA)

An RA is an entity that is responsible for identification and authentication of the subjects of certificates, but is not a CA, and hence does not sign or issue certificates. 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.

5.5 End entities

An end entity is a certificate subject which uses its private key for purposes other than signing certificates.

6 Certification authority systems

6.1 Introduction

A CA and one or more RAs may be configured as a CA system, and the functional allocation between CAs and RAs will vary, based on implementation.

RAs may be appointed to look after the secure registration of entities or this function may be part of the services provided by the CA. Each RA is responsible for a specific RA domain.

When an RA is used, communications between the RA and the CA shall be authenticated. A digital signature may be used for this process

6.2 Responsibilities in CA systems

6.2.1 Sole responsibilities and best practices of a CA

6.2.1.1 Sole responsibilities of a CA

The CA shall be solely responsible for

a) securing the private key associated with the public key contained in the CA's certificate,

b) ensuring that public keys certified by the CA are unique within the CA's domain,

c) ensuring that there is no duplication of the requester's distinguished name with that of any other entity certified by the CA, [see 7.4 (2) and 7.5 (6)],

d) generating certificates by applying a signature to the certificate information,

e) maintaining a revocation checking mechanism appropriate for relying parties,

f) creating, maintaining and distributing certificate revocation lists, and

g) changing the CA's public/private key pair at intervals appropriate to the business risk.

A CA system shall ensure that the responsibilities of the CA and RA are assigned.

The operation of a CA shall be in accordance with prudent business practices and the CA's certificate policy and certification practice statement (CPS).

The CA shall be responsible for certification and shall be able to print the information contained in certificates in human readable form.

Certification path constraints are defined in ISO 15782-2. Examples of their use are given in Annex K of ISO 15782-2.

Distribution requirements for a CA's public key are defined in 6.3.5.

6.2.1.2 Best practices of a CA system

It is recommended that the CA system possess the following attributes.

a) It should have sufficient resources to maintain its operations in conformity with its duties.

b) It should be reasonably able to bear its risk of liability to end entities and persons relying on certificates issued by the CA, as dictated by its policy.

c) It should employ personnel practices which provide reasonable assurance of trustworthiness.

d) It should use standardized (ISO or national) cryptographic techniques and cryptographic modules designed to meet the requirements for financial institution use based on four levels of module security defined as follows.

1) 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.

2) 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.

3) 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.

4) 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.

NOTE These levels are not to be confused with ISO/IEC 15408's 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.

e) It should make its own certificates, public key and certificate revocation lists (CRL) readily and reliably available to relying parties.

f) It should utilize trustworthy systems in performing its services.

g) It should define and document policy for the CA system domain.

6.2.2 Responsibilities to be allocated to either a CA or an RA

The following requirements are applicable to the CA or RA.

a) It shall validate the identity of the entity requesting the certificate as being that of the subject of the certificate.

   This may be accomplished by having the entity sign the request for the certificate and having the CA or RA validate that signature using the public key presented for certification. See 7.4.

b) It shall validate the identity of the entity requesting a certificate if the requesting entity will not be the subject of the certificate.

c) It shall if appropriate, advise the party identified in the certificate that a certificate has been issued.

   A validated means shall be used to convey this advice, and that means shall be independent of any method used to convey the certificate to the entity. Examples of validated means include normal mail for low-risk (retail) systems and registered mail for private banking systems. The implementation of this functionality is determined by business risk.

d) It shall keep records supporting the certificate issuance process for the length of time determined by records retention requirements.

e) It shall register authenticated entities securely.

f) It shall provide guidance to its end entities on the secure management of the end entity's private key.

g) It shall inform the end entity that the integrity of the end entity's operation will be considered compromised if the private key of the end entity is ever revealed to or used by any unauthorized entity.

h) It shall use any appropriate means to ascertain that the end entity understands the responsibilities of 6.2.3. and is able to comply with them.

i) It shall inform relying parties in the domain when the CA private key has been compromised.

j) It shall handle certificate revocation requests from entities.

Annex H contains suggested requirements for the acceptance of certificate request data. See 6.4 for security quality assurance and audit requirements.

6.2.3 Responsibilities of an end entity

The following requirements are applicable to the end entity.

a) It shall ensure that the end entity's private key is

   — kept within the secure confines of a cryptographic module, and

   — used only by the authorized person with proper access controls (e.g. user password or PIN).

b) It shall understand the requirements for business continuity as specified in the CPS.

c) It shall ensure that the private key, when stored in a backup environment, is kept within the secure confines of the cryptographic module, where the cryptographic modules are kept in secure storage.

Security requirements for the backup of private keys should be based on 6.3.1.1, except that Level 2 cryptographic modules may be used. In low-risk applications, a Level 1 cryptographic module may be used. The private key shall be exported in a secure manner.

NOTE Both Levels 1 and 2 of key management allow a single individual access to an un-split encryption key. This practice may be acceptable in certain perceived "low-risk" situations (such as home banking), but may not be acceptable within commercial financial institutions, which have traditionally required dual control of keying material.

d) It shall undertake that the private key is kept under its exclusive control, with due care being exercised to prevent unauthorized use throughout the life of the key.

e) It shall undertake that the CA System is notified as soon as possible if the end entity knows or suspects that its private key has been lost, disclosed, revealed or otherwise compromised.

f) It shall ensure that, following the compromise or withdrawal of an end entity's private key, the use of that private key is immediately and permanently discontinued.

g) It shall ensure that any end entity private key, that has been previously lost or compromised and later recovered, will be destroyed.

h) It shall ensure that if the end entity ceases to exist, there exists a designated responsible party which will ensure that the end entity's key is destroyed and which will notify the CA System.

i) It shall provide the level of security required by the CPS and in accordance with an end entity agreement.

6.3 Certificate life cycle requirements

6.3.1 Generation

6.3.1.1 General

This clause addresses the requirements for the generation of public/private key pairs.

6.3.1.2 Security requirements for a CA's certificate signing private key

Since the certificate generated by a CA shall be used to provide proof of the identity and integrity of the entity's public key, that part of the CA which performs private key cryptographic operations shall be implemented in a cryptographic module not controlled or accessed by any subscribing entity. Since the cryptographic module employs the private key of the CA that issues the entity certificate, this key shall be given a high level of protection, as its possession would enable an intruder to masquerade as the CA and generate forged certificates.

The security requirements for a CA's signing private key are the following.

a) The private key shall be generated internally to a cryptographic module that, as a minimum, meets the requirements of a Level 3 cryptographic module.

b) The public/private key generation process shall ensure that the keying material is arithmetically consistent with the requirements of a valid public key for the algorithm specified.

c) The public/private key pair shall be at least as strong as any key that it will be used to certify.

d) Neither the private key, nor any part thereof, shall exist in plain text outside of a cryptographic module that, as a minimum, meets the requirements of a Level 3 cryptographic module.

e) A CA shall have exclusive control over its own private key.

f) If copies of the CA's certificate signing private key are required, the private key shall be securely exported using one of the following methods:

— under at least dual control with split knowledge; or

— encrypted using a key of at least equivalent strength of the public/private key pair which itself is afforded the same high level of protection as a CA signing private key.

g) After key expiry all copies of the CA private key (and fragments if they exist) must remain securely protected or be securely destroyed.

6.3.1.3 End entity key pair generation requirements

The generation of an end entity's public/private key pair shall occur in at least a Level 1 cryptographic module. Key pair generation for high-risk applications should occur in at least a Level 2 cryptographic module. Security requirements for the backup of private keys should provide the same level of trust as for the operational private key.

Generation of the end entity's key may take place in the end entity cryptographic module or in a central key management infrastructure. It is strongly recommended that signature keys be generated in the end entity's cryptographic module. The following additional requirements apply if the key pair is centrally generated.

a) The central key generation authority shall generate the key inside at least a Level 3 cryptographic module.

b) The central key generation authority shall ensure that the end entity digital signature private key is not disclosed to any entity other than the owner of the key.

c) The central key generation authority shall not maintain a copy of any digital signature private key, once that key is delivered to the end entity.

d) The use of centralized key generation requires a secure channel to deliver the key pair to the end entity. The delivery of the end entity's private key requires confidentiality and integrity protection.

EXAMPLE 1 Delivery in a Level 1 cryptographic module under dual control with split knowledge as defined in 6.3.5 or other appropriate International Standard on banking.

EXAMPLE 2 Delivery under dual control with split knowledge.

EXAMPLE 3 Delivery in a PIN Mailer for low-risk systems.

The end entity may need to interact with the certificate management system to prove possession of the private key corresponding to the public key in the request using the process known as key validation.

6.3.2 Submission of certificate request data

An end entity shall compile its certificate request data and submit it to an RA. The certificate request data provides the RA with sufficient information to allow the RA to verify the identity of the end entity. This includes the distinguished name of the end entity.

The certificate request process shall use the entity's private key to sign the certificate request data, and the RA or CA shall validate the signature on the certificate request data to ensure

a) the integrity of the entity's distinguished name, public key, and other information during the application process using the private key,

b) that the public key in the certificate request data corresponds to the entity's private key, and

c) that there has been no failure in the key generation and transmission process.

Subsequent to key generation and prior to key usage, the end entity shall

— compile the certificate request data, including the end entity's distinguished name and newly generated public key,

— digitally sign the certificate request data using the private key that relates to the public key contained in certificate request data, and

— submit the signed certificate request data and other information as required by the CPS to an RA or a CA.

See 7.4 for a format for the submission of certificate request data.

6.3.3 Registration

When the RA or a CA performing RA functions applies for a certificate on behalf of the end entity, the CA or RA shall

a) validate the identity of the requesting end entity, according to the CPS of the CA (suggested requirements for accepting certificate request data are contained in Annex H — see also 7.4.),

b) validate the end entity's possession of the private key corresponding to the public key for which a certificate is requested,

c) accept certificate request data from the end entity whose identity has been validated (if the certificate request data was generated by the end entity),

d) check the certificate request data for errors or omissions,

e) validate the end entity's digital signature on the certificate request data,

f) deliver a copy of the CA's public key in accordance with the requirements of 6.3.5 and a copy of the end entity's certificate to the end entity,

g) provide a notification to the end entity confirming successful registration and issuance of the certificate using an out-of-band method, and

h) depending on business, record its actions in an audit journal.

See 6.4.1 and Annex F.

Figure 2 summarizes a process for issuing a certificate using a CA alone. Figure 3 summarizes one approach for issuing a certificate by a CA using an RA.

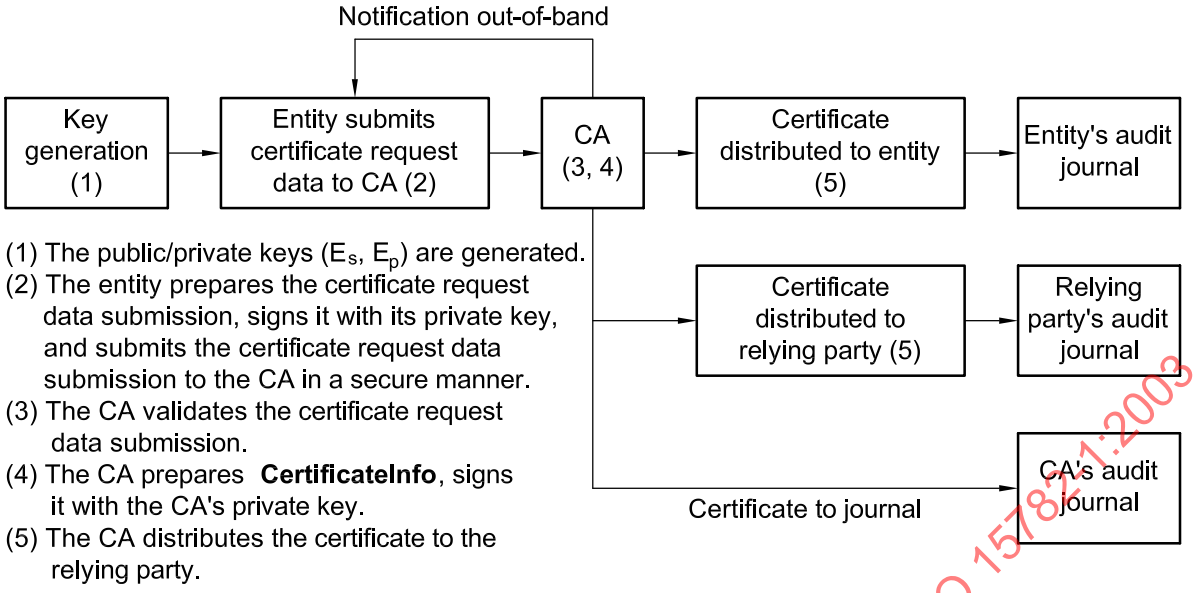


Figure 2 — Issuance of certificate by CA alone

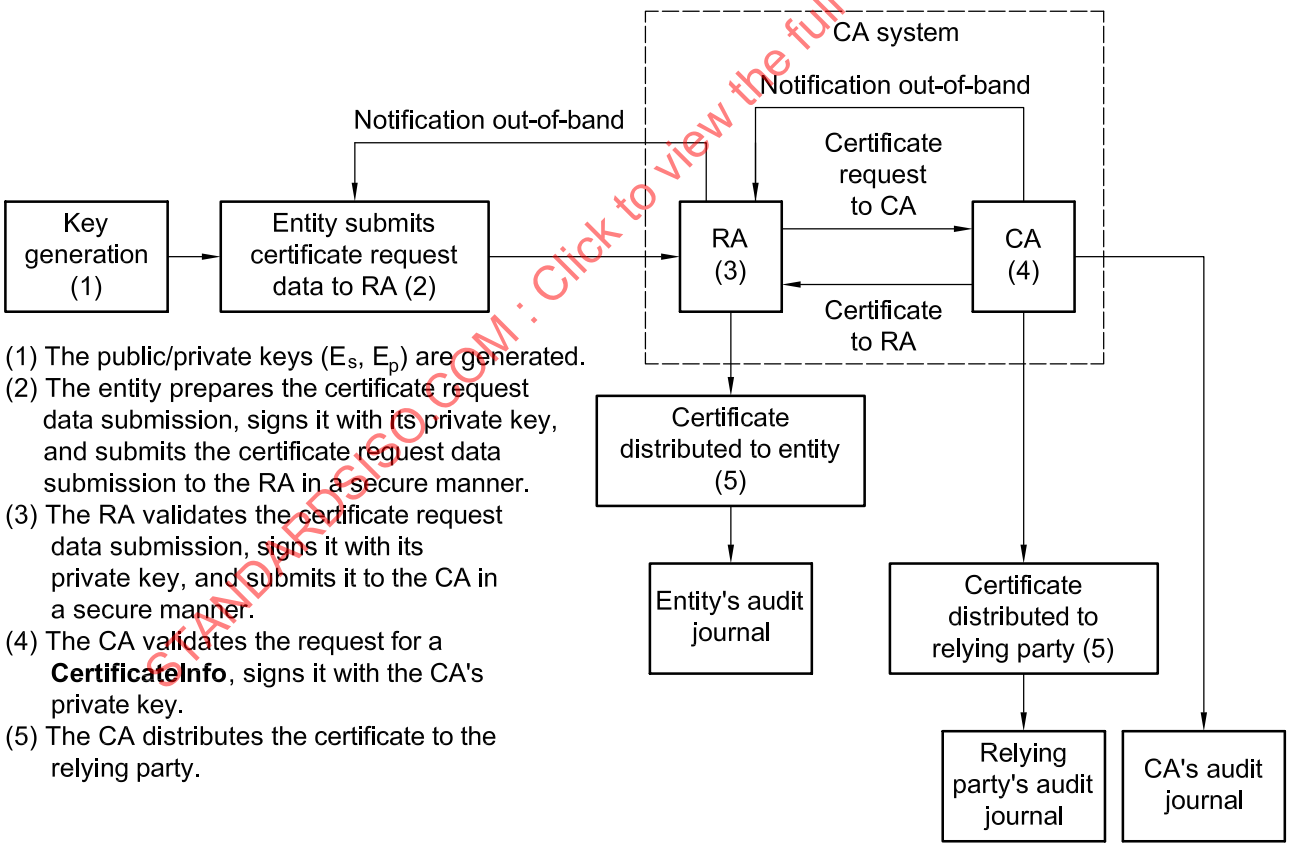


Figure 3 — Issuance of certificate by CA using an RA

6.3.4 Certification

For requests for new public key certificates (X.509, v3), the CA shall

a) verify the authenticity of the submission by the RA,

b) validate the signature on the certificate request data submission,

c) ensure that there is no duplication of the requester's distinguished name with that of any other entity certified by the CA,

d) ensure the uniqueness of the public key submitted for certification within the CA's domain, and,

e) when duplicate public keys are detected, the CA shall

   — revoke all certificates that contain the duplicated public key; and

   — reject the request for a certificate.

CA Systems shall perform public key validation based on business risk. If any of these checks fail, the CA shall reject the certificate request.

For certificate requests, the CA shall

— ensure that the elements of the certificate information provided in the certificate request data are in compliance with the CA's CPS and, if appropriate, complete or modify those elements to achieve compliance (see 7.4),

— generate or obtain additional data elements to complete the certificate information (**CertificateInfo**), depending on the type of certificate requested,

— use one or more CA private keys to sign the **CertificateInfo,** thereby creating one or more certificates depending upon the business model in operation,

— verify its own signature on the certificate prior to issuing the certificate,

— deliver a copy of the CA's public key or keys in accordance with the requirements of 6.3.5 and a copy of the end entity certificate(s) to the RA[1)] or directly to the end entity, depending on the business model,

— securely notify the RA, using an out-of-band means if necessary, that one or more certificates have been generated, and

— record its actions in an audit journal.

6.3.5 Distribution of CA public keys

The CA system shall provide its entities and relying parties with one or more trusted CA public keys and associated parameters, which these entities can use to validate the first certificate in a path.

The CA may distribute multiple public keys to provide for the replacement of a public key upon the expiration of the cryptoperiod of a given public/private key pair and for backup and recovery purposes.

1) The RA does not need to receive the CA public key on every request and thus can provide the CA public key to the subject at the time of submission.

The integrity of a CA's public key and any associated parameters is essential. When a single CA or a certification path is employed, the integrity of the certification path depends on the integrity of the public key and the confidentiality of the private key of every CA in that path.

The CA shall

— distribute its public key, associated parameters including the validity period and the CA's distinguished name, and

— ensure the integrity and authenticity of that key during distribution.

The method of distribution will depend upon the business risk model employed, and the reason for distribution, but shall be one of the following.

a) Trusted CA public keys may be initially distributed using

   — machine readable media (e.g. IC card),

   — embedding in an entity's cryptographic module,

   — a self-signed certificate, which contains the trusted public key, signed with the corresponding private key (note that the signature does not in itself prove the identity of the signer, this being done via out-of-band means), or

   — non-automated means.

b) if a subscriber or relying party already has an authenticated copy of a trusted CA public key, a new trusted CA public key may be distributed by

   — direct electronic transmission from the CA,

   — placing into a remote cache or directory,

   — loading into a cryptographic module, or

   — any of the methods for initial distribution.

Regardless of the method of transmission, the integrity and authenticity of the trusted CA public key shall be ensured. This may be achieved using one of the following, or an equivalent method.

— Distribute media under dual control with split knowledge and obtain receipts. This method is not applicable to direct electronic transmission.

— Sign a new trusted CA public key using an existing trusted CA private key. If this method is used, the recipient shall validate the signature on receipt.

— Where still used, a message authentication code (MAC) as defined in an ISO standard may be used to provide integrity protection for trusted CA public keys distributed to a database or local cache.

If this method is used, the private key used to generate the digital signature or the secret key used to compute the MAC shall not be used for any other purpose.

For high-risk applications, either the triple-DES MAC defined in ISO 9807:1991, Annex C, or single-DES MAC with each database or cache entry signed with a different key shall be used. For low- and medium-risk applications, a single-DES MAC with keys managed using any approved TC 68 key management standard is sufficient.

A CA's key may be contained in a certificate signed by another CA and be a part of a certification path, in which case the key may be validated using the public key from the previous certificate in the path.

To validate a path that includes multiple CA certificates, the initial certificate in a path is validated by a trusted CA public key that shall be distributed by one of the methods described in this part of ISO 15782.

The validator shall ascertain that this trusted CA public key is currently valid. Annex I describes a method for extracting a CA's public key from a cross domain certification path (a path that includes one or more CAs using two or more digital signature algorithms, e.g. DSA and RSA).

6.3.6 Distribution of certificates

Certificates may be distributed to relying parties using one or both of the following methods:

— certificates are explicitly included with the electronic message;

— certificates are implicitly in the electronic message and are contained in a remote cache or repository.

The RA may also distribute the certificate to relying parties. Note that the means for distribution are not specified in this part of ISO 15782.

6.3.7 Usage of certificates

The usage of certificates shall meet the following requirements.

a) In order to validate a certificate and obtain a validated public key for immediate use, the relying party shall check

   1) the validity period of the certificates (note that synchronization and secure maintenance of the clocks of the sender, recipient and all CAs in the path is an issue and that solutions should be based on business risk).

   2) the revocation status of all certificates in the certification path, using CRLs or other appropriate status mechanisms (e.g. Online Certificate Status Protocol (OCSP), or other standard certificate status protocol), and

   3) the validity of the signatures on all certificates in the certification path:

      — If the system uses a trusted time mechanism and the certificate was revoked *before* the trusted time parameter contained in the signed message then the certificate shall be treated as invalid, and any business decision that contradicts this requirement will be outside the scope of the certification system and will implicitly accept the possibility that the message may be repudiated, or

      — if the system does not use a trusted time mechanism, then it may be possible to validate signed messages using a certificate that has been revoked but that appears to have originated before the revocation took place.

   A business decision shall be made by the relying party as to whether to accept or reject the message. This will depend upon the business risk profile of the system and will take into account the possibility that the message may be repudiated.

b) Messages that fail path validation for any reason shall be rejected:

   1) for high-risk applications, relying parties shall record validation failures in an audit journal;

   2) relying parties shall retain records associated with validation failures for the period of time required by law, regulation, or prudent business practice, and these records shall include

— the message that failed validation,

— the certificate chain associated with the message, and

— the certificate revocation lists (CRL) or the use of other standard certificate status checking protocol.

A certificate shall be used only for the purposes outlined in the certificate policy.

See ISO/IEC 9594-8 | ITU-T Recommendation X.509 (1997) and ISO 15782-2 for detailed certificate path processing requirements.

The integrity of any public key and associated parameters retained for future use shall be ensured. One method is to store the entire certificate and validate the certificate as required. Another method is to extract the public key for operational use, while protecting the key from accidental or deliberate modification. Regardless of the method, the relying party shall allow re-validation of the public key. This includes the validation of the time span during which the key is authorized for use and any possible certificate revocations or holds.

For accordance with 6.3.3 and 6.3.4, it is required that an entity be provided with either

— the public key of the CA which signs the entity's certificate, or

— the public key of any CA in a possible certification path between the originator and receiver of signed information.

If the public key of the relying party's CA is used to unwrap a certification path, the path extends from the relying party's CA to the originator's CA to the originator.

If the public key of another CA in this path is trusted, the path may be shortened because fewer certificates would need to be validated. For example, the trusted public key of the topmost CA in a hierarchy could be used, allowing the path to extend from that topmost CA to the originator, with no need for a path from the relying party to the topmost CA. Some examples of these alternative trust models are presented in Annex G.

6.3.8 Revocation and hold or expiry

6.3.8.1 General

A certificate has a lifetime that is indicated by a validity period stated in the certificate but may be further constrained by the CA as defined in the certificate policy.

A CA may place a temporary hold on a certificate without permanently revoking it. Reasons for such an action include

a) a desire to reduce liability for erroneous revocation when a revocation request is not authenticated and there is inadequate information to determine whether the revocation request is valid, and

b) other business needs, such as temporarily disabling the certificate of an entity pending an audit or investigation.

A certificate revocation or hold may be requested on an entity's certificate by that entity or by an RA associated with that entity. RA association is established at the time of a certificate request according to CA and RA policy. Where the prevention of denial of service is required, the revocation or hold request shall be authenticated. In this case, a digital signature may be used.

Certificates may be revoked or held only by the CA that issued the certificate. This may occur for a number of reasons:

— end entity private key compromise;

— CA private key compromise;

— change of affiliation (e.g. to a different CA);

— a public key being superseded with another;

— cessation of operations;

— certificate hold;

— unspecified reasons.

However, the reason code CRL entry extension should be absent instead of using the **unspecified** reason code value.

A CA (or RA) shall validate the identity of an entity requesting the revocation or holding of a certificate, according to commercially reasonable procedures corresponding to the relative risk of an unauthorized revocation or hold.

A procedure and means of rapid communication shall be in place to facilitate the secure and authenticated revocation or holding of

— one or more certificates of one or more entities,

— the set of all certificates issued by a CA based on a single public/private key pair used by a CA to generate certificates, and

— all certificates issued by a CA, regardless of the public/private key pair used.

Whether certificates expire, are revoked or are held, copies of old certificates and CRLs shall be retained by the issuing CA for the period of time required by law, prudent business practice and regulations. Note that this is a records retention requirement. Requirements for keeping CRL entries for expired certificates are stated in 6.3.8.2.

In the case of revocation of a certificate, all certificates containing the same public key shall be immediately revoked.

6.3.8.2 Certificate revocation process

Certificates that have been revoked or held may be distributed in time-stamped certificate revocation lists (CRLs). Time-stamping is critical for indicating the time at which the binding between an entity's public key and identity has been terminated or (held). Upon revocation, the entity whose certificate is revoked cannot generate signatures with the private key corresponding to the public key in the revoked certificate, which can be verified to be valid.

When certificate revocation lists are used, these lists shall be

— created and signed by the CA so that relying parties can validate the integrity of the CRL and the date of issuance,

— issued by the CA at regular intervals, even if no changes have occurred since the last issuance, and

— accessible to all entities and relying parties of the system.

The frequency and timing of CRL issuance is defined in the CPS. However, the on-line distribution of CRLs need not be made to all entities. CRLs may also be made available by a Directory Service. At a minimum, CRL entries identifying revoked certificates shall remain on the CRL until the end of the validity period of that certificate.

See ISO/IEC 9594-8 | ITU-T Recommendation X.509 (1997) and ISO 15782-2 for a discussion of ways to partition CRLs (e.g. delta CRLs and distribution points).

6.3.8.3 Certificate hold and release

A certificate may be placed on hold by issuing a CRL entry with a reason code of **certificatehold**. An optional hold instruction code may also be included to convey additional information to relying parties.

The duration of a certificate hold will vary according to the purpose of the hold. In the case of an unauthenticated revocation request where the CA is unsure of the identity of the requester, the hold might last for a few days or even only a few hours, as needed to investigate and validate the authenticity of the request. If the certificate of a substantial enterprise is placed on hold, the CA will wish to complete its investigation as quickly as possible. Holds issued for other business purposes might apply for a longer time.

Once a hold has been issued, the hold may be handled in one of three ways:

a) remain on the CRL with no further action, causing validation of the certificate to fail during the hold period;

b) be replaced by a revocation, for the same certificate, in which case the reason shall be one of the standard reasons for revocation and the optional instruction code extension field shall not appear;

c) be released and the entry removed from the CRL.

The certificate may be used after the expiry of the hold. The hold is released by not putting the certificate on the next CRL.

6.3.8.4 Actions taken when a certificate is revoked or held

6.3.8.4.1 General

The actions that shall be taken when a certificate is revoked or held are defined in Table 1. Additional actions to be taken for each reason code are defined in subsequent tables.

6.3.8.4.2 Actions taken during certificate revocation

During the certificate revocation process, the CA shall

a) verify the identity and authority of the entity requesting revocation of a certificate,

b) validate the revocation request,

c) prepare the revocation notice, sign it with its private key and may send it to the requesting entity,

d) prepare the **CRLInfo** and sign it with its private key,

e) ensure availability to all entities of information on certificate status, and

f) record its actions in an audit journal.

The CA should provide an authenticated acknowledgement of the revocation to the entity, and may also distribute the revocation notice to other subscribers and relying parties.

When the RA assists an entity in the certificate revocation process, the RA shall

— verify the identity and authority of the entity requesting revocation of a certificate,

— submit certificate revocation requests to the CA in an authenticated manner,

— receive and verify the confirmation that the CA has received the revocation request,

— secure that part of the revocation process for which the RA assumes responsibility,

— provide an authenticated acknowledgement of the revocation to the requesting entity,

— ensure availability to all subscribers and relying parties of information on certificate status, and

— record its actions in an audit journal.

It should also provide an authenticated acknowledgement of the revocation to the entity.

When the RA assists a requesting entity in the certificate hold process, the RA shall

— verify the identity and authority of the entity requesting the hold of a certificate,

— submit the certificate hold requests to the CA in an authenticated manner,

— provide a written acknowledgement of the hold.

See, also, Table 1.

6.3.8.4.3 Additional actions to be taken

See Tables 2 to 6.

Further additional actions that may be taken by an entity or RA requesting a hold are the following.

— An entity or RA may request that the CA that issued a certificate to be held place a hold on the certificate giving the **CertificateSerialNumber** to identify the certificate, and an optional **CRLEntry reasonCode** value of **certificateHold**.

— If the certificate to be held is that of a CA, the Certificate Revocation List of the CA itself shall contain entries for all revoked and held certificates.

Table 1 — Actions to be taken whenever a certificate is revoked or held for any reason

Entity	Actions to be taken
Certified entity or RA	The certified entity or RA may 1. request that the CA revoke, hold, or release the hold on a certificate, giving the **CertificateSerialNumber** to identify the certificate, and the optional **CRLEntry ReasonCode** requested, 2. send a message to notify other entities and relying parties and identify the certificate for the **CRLEntry**, 3. update the audit journal to reflect the actions taken and the reasons for the actions. Revoked or held certificates shall be journalized.
CA	The CA shall ascertain the validity of the revocation or hold request, according to the CA's CPS and perform the following actions. 1. Update the CRL. In the case of a certificate revocation, the certificate shall remain on the revocation list until the first CRL issued following the expiration date of the certificate, as a minimum. 2. Send (optionally) a signed message (out-of-band) containing the CRLEntry to all entities and relying parties. A revocation notice may be used for this purpose. See 7.8. 3. Update the audit journal to reflect the actions taken and the reasons for the actions. Revoked or held certificates shall be journalized.
Users of the certificate	The user shall: 1. reject any message signed after the revocation date requiring the use of the revoked certificate, 2. update the audit journal to reflect the actions taken and the reasons for the actions. Revoked or held certificates shall be journalized. Optionally, other entities may be notified. Table 6 defines additional requirements when certificates for public keys that are used to protect symmetric algorithm key exchanges are revoked or held keys used to protect symmetric algorithm key exchanges are revoked.

Table 2 — Additional actions taken on the compromise or suspected compromise of an entity's private key

Entity	Additional actions to be taken
Certified entity or RA	The certified entity or RA may request that the entity's CA revoke the certificate, giving the **CertificateSerialNumber** to identify the certificate and with a optional **CRLEntry reasonCode** value of **keyCompromise** or **caCompromise**. If the entity is a CA, If the entity is a CA, all the suspected certificates shall be revoked, and the **CertificateRevocationList** of the CA itself may contain entries for all suspect certificates, with an optional **reasonCode** value of **caCompromise**.
CA	The issuing CA shall update the Certificate Revocation List (**CertificateRevocationList**). The **CRLEntry** in the **CertificateRevocationList** may optionally contain a **reasonCode** value of **keyCompromise** or **caCompromise**, as appropriate.
Users of the certificate	Discontinue the use of all keying material ever sent and protected by that certificate.

Table 3 — Additional actions taken because of cessation of operations

Entity	Additional actions to be taken
Certified entity or RA	The certified entity or RA may request that the entity's CA revoke the certificate, giving the **CertificateSerialNumber** to identify the certificate and a **CRLEntry** and optional **reasonCode** value of **cessationOfOperation**. If the entity is a CA, it shall revoke all certificates that the CA has issued. The request may be submitted by the entity or its legal representative.
CA	The issuing CA shall update the certificate revocation list (**CertificateRevocationList**) giving the optional **reasonCode** value of **cessationOfOperation**. The business issues regarding the management and ownership of the entity or CA that has ceased operations shall be addressed.

Table 4 — Additional actions taken because of change of affiliation of entity

Entity	Additional actions to be taken
Certified entity or RA	The certified entity or RA may request that the entity's CA revoke the certificate giving the **CertificateSerialNumber** to identify the certificate with a optional requested **CRLEntry reasonCode** value of **affiliationChanged**. If the entity is a CA, the **CertificateRevocationList** of the CA itself may contain entries for all revoked certificates.
CA	The CA shall update the certificate revocation list (**CertificateRevocationList**), giving the optional **reasonCode** value of **affiliationChanged**.

Table 5 — Additional actions taken when certificates are revoked for reasons other than for key compromise, cessation of operations or change of affiliation

Entity	Additional actions to be taken
Certified entity or RA	The entity or RA may request that the entity's CA revoke the certificate giving the **CertificateSerialNumber** to identify the certificate and an optional certificate revocation list entry (**CRLEntry**) **reasonCode** value of **superseded** or **unspecified**. If the entity is a CA, the **CertificateRevocationList** of the CA itself shall contain entries for all revoked certificates, with a **reasonCode** value of **superseded** or **unspecified**.
CA	The CA shall update the certificate revocation list (**CertificateRevocationList**) giving the optional **reasonCode** value of **superseded** or **unspecified**, as appropriate.

Table 6 — Additional actions taken when certificates for public keys used to protect symmetric algorithm key exchanges are revoked or held

Reason for revocation or hold	Additional actions to be taken by users of the certificate
Compromise or suspected compromise of an entity's private key	The use of all keying material ever sent and protected by that certificate (without regard to type) shall be discontinued. If the entity whose certificate is revoked or held is a CA, a different CA shall be used in the process of replacing keying material.
Certificate expires or is revoked for reasons other than actual or suspected compromise	Replace all keying material sent and protected by that certificate (without regard to type) as soon as is operationally convenient.
Certificate is held for reasons other than actual or suspected compromise	When a certificate is held for reasons other than actual or suspected compromise, optionally suspend the use of, or replace, all keying material sent and protected by that certificate (without regard to type) as soon as operationally is convenient.

6.3.9 Renewal of certificates

A certificate has a lifetime indicated by the validity period stated in the certificate.

Before this validity period has expired, an end entity may request the renewal of a certificate by requesting an extension of its validity period (i.e. requesting a **notAfter** date later than the existing certificate's **notAfter** date). However, the start date (**notBefore** date) must be the same in the renewed certificate as it is in the original certificate.

Certificates may be renewed only by the CA that issued the certificate.

An end entity requesting the renewal of a currently valid certificate shall compile its certificate renewal data and submit it to the RA, or the CA performing RA functions, that had previously generated this certificate. The certificate renewal data provides the RA with sufficient information to allow the RA to verify the identity of the end entity and to identify the certificate to renew. This includes

a) the distinguished name of the end entity,

b) the serial number of the certificate, and

c) the requested validity period.

The end entity shall digitally sign the certificate renewal data.

See 7.7 for the format of certificate renewal data.

For renewing public key certificates the CA shall

a) validate the signature on the certificate renewal data submission,

b) verify the existence and validity of the certificate to be renewed,

c) verify that the request of the validity period includes the same **notBefore** date as the **notBefore** date in the original certificate, and that the **notAfter** date is later than the **notAfter** date in the original certificate, and

d) verify that the request, including the extension of the validity period, meets the requirements defined in the certificate policy.

If none of these checks fails, the CA shall generate and sign a new instance of the certificate, differing from the previous certificate only by the validity period, certificate serial number and the CA signature. If a check fails, then the CA shall reject the certificate renewal request.

The renewed certificate shall be distributed by the CA following the same procedure used after a certificate generation.

A certificate renewal results in two certificates with the same public key. This implies that

— renewal of a certificate does not require revocation of the previous instance of the certificate, and

— when the validity periods overlap, any of the different instances of a certificate may be used.

6.4 Security quality assurance and audit requirements

6.4.1 Introduction

CAs and RAs shall maintain sound management and control practices confirmed via security quality assurance processes and procedures, and compliance audits.

6.4.2 Audit journal requirements

CA systems are required to keep audit journals that provide sufficient detail to reconstruct events, provide "due care" requirements and meet legal requirements. All entries in audit journals shall be date and time stamped. While audit journals will normally be created and maintained by the CA management system, some audit journals may out-of-necessity be manual. The audit requirements shall be specified in the CA's certification practice statement.

CA audit journal entries shall include all certificate and key management operations, such as key generation, backup, recovery and destruction, together with the identity of the person authorizing the operation and persons handling any key material (such as key fragments or keys stored in portable devices or media). Changes in the custody of private keys and associated parameters, and of devices or media holding keys shall be recorded in the audit journals. Audit journals shall not record the plain text values of any private keys but may hold hash values as a means of identifying keys and validating their correctness as well as that of public keys derived from private keys by means of a one-way function.

A list of audit journal contents is provided in Annex F.

Audit journals shall be maintained in a form that prevents unauthorized modification or destruction. Automated audit journals shall be protected from modification or substitution. The use of a hash and a digital signature may be used as defined in ANS X9.30, ANS X9.31 and ANS X9.62. The private key pair used for signing the audit journal shall not be used for any other purpose. In addition, the audit journal shall only be retrieved by authorized individuals for valid business or security reasons.

6.4.3 Security quality assurance

Documented security quality assurance processes and procedures are required as part of the system of internal security control over certificate management. The audit journal shall be reviewed regularly (e.g. daily) by an security quality assurance function. In some organizations, this function may be fulfilled by the audit department. The review shall include the validation of the audit journal's integrity, and the identification and follow-up of exceptional, unauthorized or suspicious activity (e.g. digital signature failures, access at unusual times or from unusual sources, unexpected increases in volume or saturation of system resources).

The extent and frequency of review and management escalation requirements should be determined by a threat/risk evaluation. In high-risk applications or for legal purposes or both, CA systems may require end entities and relying parties to maintain an audit journal.

6.4.4 CA and RA audit

Compliance reviews shall be performed by an independent audit function (internal, external or both) at a frequency (e.g. annually) to be determined by the audit function. The audit should include compliance with the certificate policy and certification practice statement, procedures manuals and configuration specifications. Audit results shall be formally documented and provided to the auditee for follow-up.

6.4.5 End entity audit

In high-risk applications, end entities and relying parties may also be audited.

6.5 Business continuity planning

The requirements for business continuity planning depend on the availability and continuity needs of the business applications supported by the CA. In general, high-risk, high-availability applications require more robust business continuity processes and techniques than low-risk, low-availability applications.

At a minimum, business continuity planning shall include disaster recovery processes for all critical components of a CA system, including the hardware, software and keys, in the event of a failure of one or more of these components.

Disaster recovery options may include the re-installation of the CA system and the re-issuance of all certificates, the use of a fully redundant system, or a "hotsite".

Disaster recovery processes are also required if a critical security component is compromised. In particular, the compromise or suspected compromise of a CA's private key shall be considered a disaster. All certificates signed with that key after the compromise date (where available) shall be considered suspect and therefore be revoked. The security measures implemented at the CA in order to protect the private key of the CA shall ensure that the probability of a compromise of that key is negligible.

In the event that a CA has to replace or recover its private key, then procedures shall be in place for the secure and authenticated cancellation of

— the set of all certificates issued by a CA based on a single asymmetric key pair, and

— all certificates issued by a CA, regardless of asymmetric key pair used.

Two possible disaster recovery procedures are described in Annex J.

An entity may have one or more valid certificates in anticipation of a need for transition or recovery. These certificates provide continuity of service when a certificate expires, a cryptographic module fails or a private key is compromised.

7 Data elements and relationships

7.1 Introduction

The authentication framework defined in this part of ISO 15782 provides for either a hierarchical or non-hierarchical structure for point-to-point connections and is based on ISO/IEC 9594-8 | ITU-T Recommendation X.509 (1997). The authentication framework is based on providing assurances that the public key of an entity is contained in the certificate request data of that entity. In turn, this assumes that there are assurances that the entity presenting certificate request data to be signed is the entity holding the private key corresponding to the public key contained in the certificate request data. ASN.1 is used in the definition of the authentication framework.

7.2 Public keys

Public keys shall be encoded in accordance with Annex E.

7.3 Signatures

Signatures are defined by reference to the **SIGNED** parameterized type, defined as a **SEQUENCE** of the data being signed, an algorithm identifier and a bit string which is the actual signature. A public key signature algorithm is applied to the ASN.1 DER-encoded representation of the data to be signed.

```
SIGNED { ToBeSigned } ::= SEQUENCE {
        toBeSigned    ToBeSigned,
        algorithm AlgorithmIdentifier {{SignatureAlgorithms}},
        signature BIT STRING
}
```

7.4 Certification request data (CertReqData)

The **CertReqData** includes that information which the entity shall communicate to the CA or RA in order that a public key certificate can be generated. The **CertReqData** is encoded and signed by the requesting entity to create a **CertReqDataSubmission** which is then submitted to the CA or RA. The format for a **CertReqDataSubmission** certification request is as follows.

```
CertReqDataSubmission ::= SIGNED { EncodedCertReqData }

EncodedCertReqData ::= TYPE-IDENTIFIER.&Type( CertReqData )

CertReqData ::= SEQUENCE {
        version             Version  DEFAULT v1,
        subject    Name          Name,
        subjectPublicKeyInfo    SubjectPublicKeyInfo,
        requestedPeriod    [0] Validity  OPTIONAL,
        timeStamp           Time  OPTIONAL,
        extensions          [2] Extensions { CertExtensions }  OPTIONAL
}

Extensions { IOSet } ::= SEQUENCE OF Extension { IOSet }

Extension { IOSet } ::= SEQUENCE {
        extnID              EXTENSION.&id({IOSet}),
        critical            EXTENSION.&critical({IOSet}{@extnID})  DEFAULT FALSE,
        extnValue           OCTET STRING
        -- An "octet hole", the value of extnValue contains the DER
        -- encoding of a value of type &ExtnType, an open type, for
        -- the extension object identified by extnId.
}

EXTENSION ::= CLASS {
        &id         OBJECT IDENTIFIER  UNIQUE,
        &critical   BOOLEAN  DEFAULT FALSE,
        &ExtnType
}
  WITH SYNTAX
            { SYNTAX &ExtnType [CRITICAL &critical] IDENTIFIED BY &id }

CertExtensions EXTENSION ::= {
        AuthorityKeyIdentifier      |
        basicConstraints            |
        certificatePolicies         |
        cRLDistributionPoints       |
        extKeyUsage                 |
        issuerAltName               |
        keyUsage                    |
        nameConstraints             |
        policyConstraints           |
        policyMappings              |
        privateKeyUsagePeriod       |
        subjectAltName              |
        subjectDirectoryAttributes  |
        subjectKeyIdentifier,
        ...  -- Expect other extension objects in future revisions of this part of ISO 15782--
}
```

The fields of the **CertReqData** are defined as follows.

a) Version (**version**)

The version number differentiates between different versions of **CertReqData**. For this part of ISO 15782, the version number shall be **v1** (see the definition in 7.5).

b) Subject name (**subject**)

This field is the name of the entity wishing to be certified. Names in certificates are defined as a sequence of components called relative distinguished names, each of which is an attribute type and value. The entity that is to be bound to the public key is contained in the **subject** field.

Entity names shall be unique. Procedures for ensuring global uniqueness can be found in ISO/IEC 9834-1:1993.

```
Name ::= CHOICE {  -- only one possibility for now --
        rdnSequence  RDNSequence

}

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::= SET SIZE(1..MAX) OF AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
        type  ATTRIBUTE.&id({RDNameAttributes}),

        valueATTRIBUTE.&Type({RDNameAttributes}{@type})

}

RDNameAttributes ATTRIBUTE ::= {
        commonName                |
        countryName           |
        organizationName      |
        organizationalUnitName,
        ...  -- Expect other implementation specific attributes  --
}

commonName ATTRIBUTE ::= {
        WITH SYNTAX         DirectoryString { ub-common-name }
        ID                  id-at-commonName
}

countryName ATTRIBUTE ::= {
        WITH SYNTAX         PrintableString( SIZE(2) )
        ID              id-at-countryName
}

organizationName ATTRIBUTE ::= {
        WITH SYNTAX         DirectoryString { ub-organization-name }
        ID              id-at-organizationName
}

organizationalUnitName ATTRIBUTE ::= {
        WITH SYNTAX         DirectoryString { ub-organizational-unit-name }
        ID              id-at-organizationalUnitName
}
```

```
ub-common-name      INTEGER ::= 64

ub-organization-name INTEGER ::= 64

ub-organizational-unit-name     INTEGER ::= 64

id-at OBJECT IDENTIFIER ::= { ds 4 }  -- Attribute Type

id-at-commonName    OBJECT IDENTIFIER ::= { id-at   3 }

id-at-countryName      OBJECT IDENTIFIER ::= { id-at   6 }

id-at-organizationName     OBJECT IDENTIFIER ::= { id-at 10 }

id-at-organizationalUnitName    OBJECT IDENTIFIER ::= { id-at 11 }
```

c) Entity's public key information (**subjectPublicKeyInfo**)

The public key of the entity to be certified is contained in the **subjectPublicKeyInfo** field.

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

SupportedAlgorithms ALGORITHM-ID ::= { … }
```

Object identifier and **ALGORITHM-ID** definitions for the algorithms used in this part of ISO 15782 are listed in Annex D.

d) Validity period (**requestedPeriod**)

Every certificate shall have a validity period. This validity period shall be identified by **notBefore** and **notAfter** dates (and times), using the **Time** structure defined below. If times are expressed as **UTCTime**, the century is derived as according to the following.

Before a value of Time is used in any comparison operation, e.g. as part of a matching rule in a search, and if the syntax of **Time** has been chosen as the **UTCTime** type, the value of the two-digit year field shall be rationalized into a four-digit year value as follows.

— If the two-digit value is 00 through 49 inclusive, the value shall have 2 000 added to it.

— If the two-digit value is 50 through 99 inclusive, the value shall have 1 900 added to it.

See ISO 9594-8 | ITU-T Recommendation X.509 (1997).

NOTE The use of **GeneralizedTime** could prevent interworking with implementations unaware of the possibility of choosing either **UTCTime** or **GeneralizedTime**. It is the responsibility of those specifying the domains in which certificates defined in this directory specification will be used, e.g. profiling groups, as to when the **GeneralizedTime** may be used. In no case shall **UTCTime** be used for representing dates beyond 2049.

The **validity** field may be provided in the certificate request and may be modified by the CA in the certificate created from the **CertReqData**. See the restrictions given in 8.2.5.

The Validity period is defined as follows:

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

Time ::= CHOICE {
        utcTime         UTCTime,
        generalizedTime    GeneralizedTime
}
```

e) Time (**timestamp**)

The timestamp field may be included to allow the CA to detect replays of old certification requests.

f) Extensions (**extensions**)

The extensions field allows the addition of new fields to the structure without modification to the ASN.1 definition. An extension consists of a unique identifier (an object ID), the data type of the extension (some ASN.1 type) and a criticality flag. If the criticality flag is **FALSE**, an implementation should ignore unrecognized extensions. If the criticality flag is **TRUE,** unrecognized extensions shall cause the structure to be considered invalid.

EXAMPLE In a certificate, an unrecognized critical extension would cause validation of a signature using that certificate to fail.

7.5 Public key certificates

The definition of a certificate that follows indicates which fields are provided by the CA. The CA may make additional information available by other means. A certificate shall have the following contents.

```
Certificate ::= SIGNED { EncodedCertificate }

EncodedCertificate ::= TYPE-IDENTIFIER.&Type( CertificateInfo )

CertificateInfo ::= SEQUENCE {
        version    [0]        Version DEFAULT v1,
        serialNumber          CertificateSerialNumber,
        signature          AlgorithmIdentifier {{SignatureAlgorithms}},
        issuer                Name,
        validity              Validity,
        subject               Name,
        subjectPublicKeyInfo    SubjectPublicKeyInfo,
        issuerUniqueID     [1]   IMPLICIT UniqueIdentifier  OPTIONAL,
        subjectUniqueID    [2]   IMPLICIT UniqueIdentifier  OPTIONAL,
        extensions         [3]   Extensions { CertExtensions }  OPTIONAL
}
```

a) Version (**version**)

The **Version** is used to differentiate between versions of the certificate.

```
Version ::= INTEGER { v1(0), v2(1), v3(2) }
```

v1 is the format defined in X.509 (1988). **v2** is the format defined in X.509 (1993), which includes the **subjectUniqueID** and **issuerUniqueID** fields. **v3** is the format defined in ISO/IEC 9594-8 | ITU-T Recommendation X.509.

b) Serial number (**serialNumber**)

The **serialNumber** field uniquely identifies this certificate among all those issued by the same CA. The **CertificateSerialNumber** shall never repeat. The combination of issuer name and serial number uniquely identifies a certificate for purposes such as the use of certificate revocation lists.

CertificateSerialNumber ::= INTEGER

c) Signature algorithm (**signature**)

The **signature** field identifies the algorithm used to sign the certificate. It is added by the CA when creating the certificate.

d) Issuer name (**issuer**)

The **issuer** field contains the distinguished name of the CA, and is added by the CA when creating the certificate.

e) Validity (**validity**)

The **validity** field indicates the period during which a public/private key pair is valid. Every certificate shall have a validity period. The validity period shall be identified by **notBefore** and **notAfter** dates. The validity period in the certificate may be different from that requested by the entity in the **CertReqData**. However, the **notAfter** time in the certificate shall never be later than the **notAfter** time in the **CertReqData**, and the **notBefore** time in the certificate shall never be before the **notBefore** time in the **CertReqData**. Note that a CA's certificate indicates the validity period of that CA's public key.

In accordance with the established security policy, appropriate mechanisms and practices shall be followed such that the UTC time is both correct and secure in order to ensure the integrity of validity period fields.

f) Name (**subject**)

This field is taken from the **CertReqData** submitted by the entity wishing to be certified.

g) Subject public key (**subjectPublicKeyInfo**)

This field is taken from the **CertReqData** submitted by the entity wishing to be certified.

h) Issuer unique ID (**issuerUniqueID**)

This optional field is used to distinguish multiple CAs with the same name. Unique ID syntax and semantics are described in 7.5. If the CA's public key is certified, then this is the subject unique ID from the CA's certificate. This field shall not be used.

i) Subject unique ID (**subjectUniqueID**)

This optional field is used to distinguish multiple entities with the same name. This field shall not be used.

j) Extensions

See 7.4.

7.6 Hold instruction codes

A certificate hold notice may contain an optional hold instruction code to convey additional information to relying parties. The following codes are defined by this part of ISO 15782:

a) **none** No instructions.

b) **callIssuer** Please communicate with the CA.

c) **reject** Locking out the certificate may be appropriate for

— scheduled non-emergency holds, or

— cases where physical actions by the relying party may be inapplicable, such as where the subject is a device or process.

d) **pickupToken** repossess user token, assuming that the relying party has acquired temporary possession of the token and the physical token is in fact the property of the issuing CA, or another entity.

Additional codes may be defined at a later date.

See Clause D.4 for the object identifiers (OIDs) for the hold instruction codes.

7.7 Certification renewal data (CertRenData)

The **CertRenData** includes the information which the entity shall communicate to the CA or RA in order that a public key certificate can be renewed. The **CertRenDataSubmission** is signed by the entity and then submitted to the CA or RA.

```
CertRenDataSubmission ::= SIGNED { EncodedCertRenData }

EncodedCertRenData ::= TYPE-IDENTIFIER.&Type( CertRenData )

CertRenData ::= SEQUENCE {
        subject         Name,
        serialNumber    CertificateSerialNumber,
        validity        Validity
}
```

The fields are defined as follows.

a) Subject Name (**subject**)

This field is the name of the entity wishing to renew the certificate [see definition in 7.4 b)].

b) Serial Number (**serialNumber**)

The **serialNumber** field uniquely identifies the certificate among all those issued by the same CA [see definition in 7.5 b)].

c) Validity (**validity**)

The **validity** fields indicate the period during which a public/private key pair is valid [see definition in 7.4 d)].

7.8 CRL data structures

The structure below is provided for information. For the definition of CRLs, see ITU-T Recommendation X.509 and ISO 15782-2.

```
CertificateList ::= CRL

CRL ::= SIGNED { EncodedCertificateList }

EncodedCertificateList ::= TYPE-IDENTIFIER.&Type(CertificateListInfo)
```

```
CertificateListInfo ::= SEQUENCE {
        version             Version  OPTIONAL,  -- if present, must be v2
        signature       AlgorithmIdentifier {{SignatureAlgorithms}},
        issuer              Name,
        thisUpdate          Time,
        nextUpdate          Time  OPTIONAL,
        revokedCertificates     CRLEntryList  OPTIONAL,
        crlExtensions       [0] Extensions { CRLExtensions } OPTIONAL
}

CRLEntryList ::= SEQUENCE OF CRLEntry

CRLEntry ::= SEQUENCE{
        userCertificate         CertificateSerialNumber,
        revocationDate      Time,
        crlEntryExtensions  Extensions { CRLEntryExtensions } OPTIONAL
}

CRLExtensions EXTENSION ::= {
        authorityKeyIdentifier      |
        cRLNumber               |
        deltaCRLIndicator       |
        issuerAltName               |
        issuingDistributionPoint,
        ...  -- Expect other extension objects in future revisions of this part of ISO 15782 --
}

CRLEntryExtensions EXTENSION ::= {
        certificateIssuer   |
        holdInstructionCode     |
        invalidityDate      |
        reasonCode,
        ...  -- Expect other extension objects in future revisions of this part of ISO 15782--
}

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

CRLNumber ::= INTEGER (0..MAX)

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)
}
```

```
holdInstructionCode EXTENSION ::= {
        SYNTAX      HoldInstruction
        IDENTIFIED BY    id-ce-instructionCode
}
        -- This extension is always non-critical --

HoldInstruction ::= OBJECT IDENTIFIER
        -- Annex C defines OIDs for the following "standard" instructions: --
        -- none, callIssuer, reject, and pickupToken. --

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

The **version** field is used to differentiate between different versions of the CRL format.

The **thisUpdate** time is the time at which the CA stopped accepting entries for this CRL. The **nextUpdate** time is the time at which the CA will issue the next CRL.

A CRL entry may contain a **reasonCode**. If this extension is not present, the entry should be processed as if it had a **reasonCode** of **unspecified**.

The CRL number is incremented by one when a CRL is issued. This allows a relying party to detect when a CRL was issued prior to the next scheduled update indicated in the **nextUpdate** field of the last received CRL.

A variety of distribution mechanisms are possible for CRLs, including delivery to each user as a message/transaction, requests by a user for the current status of a given certificate, and queries to the CA for its current **CertificateRevocationList**. In any case, the CA shall publish and distribute a new **CertificateRevocationList** on or before the **nextUpdate** time in the current **CertificateRevocationList.**

The provisions of 6.3.8 allow the CA to inform other entities of individual revocations. This may be done with a **RevocationNotice**:

```
RevocationNotice ::= SIGNED { EncodedRevNoticeInfo }

EncodedRevNoticeInfo ::= TYPE-IDENTIFIER.&Type(RevocationNoticeInfo)

RevocationNoticeInfo ::= SEQUENCE {
        signature AlgorithmIdentifier{{SignatureAlgorithms}},
        issuer          Name,
        crlEntry    CRLEntry
}
```


8 Public key certificate and certificate revocation list extensions

8.1 Introduction

ISO/IEC 9594-8 | ITU-T Recommendation X.509 (1997) provides the syntax and semantics for a public key certificate. Version 3 (V3) public key certificates provide a mechanism for CAs to append additional information about the entity's public key, issuer's public key and issuer's CRLs. Standard certificate extensions are defined. Since ISO/IEC 9594-8 | ITU-T Recommendation X.509 (1997) is intended to be applicable to a widely diverse community of users, it offers numerous optional features. This clause describes a profile of the certificate and CRL extensions that appear in ISO/IEC 9594-8 | ITU-T Recommendation X.509 (1997). This profile defines the requirements for implementing public key certificates, i.e. constructing and processing public key certificates, for financial applications. This profile also addresses similar requirements for CRLs.

While not all extensions as defined in ISO/IEC 9594-8 | ITU-T Recommendation X.509 (1997) are applicable to financial applications, all of the extensions are included here to ensure completeness. Annex C provides a tabular representation of these requirements with respect to their implementation in ASN.1.

NOTE ISO 15782-2 is extracted from Clause 12 of ITU-T Recommendation X.509 (1997), *Certificate and CRL Extensions*, adapted to meet the requirements of financial applications.

8.2 Certificate extensions

8.2.1 Authority key identifier

This non-critical extension identifies the public key used to verify the signature on a certificate. It enables distinct keys used by the same CA to be differentiated. This extension may hold an explicit key identifier or an explicit certificate identifier. This extension is useful when a CA uses more than one key (e.g. when the CA key is re-keyed). The use of this extension may provide improved efficiency when attempting to locate a specific certificate.

CAs shall be capable of generating this extension in all CA and end entity certificates. When the extension is populated, CAs shall

a) include the **keyIdentifier** octet string and

b) omit the **authorityCertIssuer** and **authorityCertSerialNumber** fields.

The **keyIdentifier** component is used exclusively because all public keys are assigned a unique identifier (see 8.2.2 and the use of the explicit key identifier will always uniquely identify a public key certificate.

There are no requirements for implementations to process this extension, though it is recommended that relying parties be capable of recognizing the **authorityKeyIdentifier** field as the identifier of the certification authority's key used to sign the certificate.

8.2.2 Subject key identifier

This non-critical extension identifies the public key being certified. It enables distinct keys used by the same subject to be differentiated. This extension may hold an explicit key identifier and is useful when a subject uses more than one key. Similar to the authority key identifier extension, this extension may provide improved efficiency when attempting to locate a certificate.

CAs shall be capable of generating this extension in all CA and end entity certificates.

There are no requirements for implementations to process this extension, though relying parties should be capable of recognizing the **subjectKeyIdentifier** field as the identifier of the public key being certified.

8.2.3 Key usage

This extension indicates the purposes for which the certified public key is used. It shall be implemented as a critical extension. The **keyUsage** field includes the following types.

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) below]. If present, the **nonRepudiation** bit may be ignored.

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 shall 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.

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 given in Table 7 and described in the following.

When the public key is used to validate the authenticity and integrity of signed data (exclusive of certificates and CRLs), then **digitalSignature** shall be set to "1".

For financial systems, non-repudiation can only be conveyed through business agreements. Therefore, these agreements may superceed the setting of the **nonRepudiation** bit. The use of the **nonRepudiation** bit is beyond the scope of this part of ISO 15782.

When the public key is used for confidentiality protection using reversible asymmetric algorithms, then only the **keyEncipherment** bit shall be set to "1". This is illustrated as Combination 2 in Table 7. The public key shall not be used for any other purposes (no other bits may be set to "1").

When the public key is used for confidentiality protection of data, then only the **dataEncipherment** bit shall be set to "1". This is illustrated as Combination 3 in Table 7. The public key shall not be used for any other purposes (no other bits may be set to "1").

When the public key is used for confidentiality protection using irreversible asymmetric algorithms, then only the **keyAgreement** bit shall be set to "1". This is illustrated in Combination 4 in Table 7. The public key shall not be used for any other purposes (no other bits may be set to "1").

When the public key is used for validating the authenticity and integrity of signed data, certificates and CRLs, then the **digitalSignature**, **keyCertSign** and **cRLSign** bits shall be set to "1", respectively. No other bits shall be set to "1". This is illustrated in Combination 5 in Table 7. A subset of this combination is also valid. The decision to combine these functions in a single public key shall be based on business risk. This combination shall only appear in CA certificates.

When the public key is used to provide confidentiality using a key agreement mechanism, e.g., KEA, Diffie-Hellman or a variant thereof, the **keyAgreement** bit is set to "1"; if the key is to be used to form a pairwise key to decrypt data, then the certificate processing entity shall ensure that the **encipherOnly** bit is set to "0"; if the key is to be used to form a pairwise key to encrypt data, then the certificate processing entity shall ensure that the **decipherOnly** bit is set to "0".

Relying parties shall interpret the **keyUsage** bits in accordance with this part of ISO 15782 and reject any combination of key usages that is not a valid combination as identified in Table 7 and described above.

Table 7 — Valid combinations of key usage bits

Key usages	Valid key usage combinations				
	End entity certificates				CA certificates
	1	2	3	4	5
digitalSignature	•	x	x	x	A
keyEncipherment	x	•	x	x	x
nonRepudiation	#	#	#	#	#
dataEncipherment	x	x	•	x	x
keyAgreement	x	x	x	•	x
keyCertSign	x	x	x	x	A
cRLSign	x	x	x	x	A
encipherOnly	x	x	x	x	x
decipherOnly	x	x	x	x	x

• Bit is set to "1".
A Any combination of these key usages is valid.
x Not allowed.
Beyond the scope of this part of ISO 15782.

8.2.4 Extended key usage

This extension indicates one or more purposes for which the certified public key may be used in addition to, or in place of, the basic purposes indicated in the key usage extension. It may be implemented as either a critical or a non-critical extension. Key purposes may be defined by any organization with a need. No additional key usages are defined by this part of ISO 15782, hence, support of this extension is optional.

8.2.5 Private key usage period

This non-critical extension indicates the period of use of the private key corresponding to the certified public key. It is applicable only for digital signature certificates. This extension is only useful when a trusted time stamp mechanism is available for comparing the date of signature of a message to the validity period included within this extension. Since no such mechanism is currently available, support of this extension is optional.

8.2.6 Certificate policies

This extension lists certificate policies that the certificate is expressly recognized as supporting, together with optional qualifier information pertaining to these policies. The certificate policies extension should be implemented as a critical extension. This field is processed in concert with the **policyConstraints** extension (see 8.2.13) during the certification path validation as described in ISO/IEC 9594-8 | ITU-T Recommendation X.509. The certificate policy indicates the procedures under which the certificate was created.

CAs shall include this extension in all CA and end entity certificates and may indicate that this extension is non-critical by setting the criticality flag to “false”. Non-criticality of this extension indicates that the use of this extension is not constrained by the policies listed. CAs shall indicate the object identifiers of applicable certificate policies. As required by the **policyConstraints** extension, it is necessary for the certificate to contain an acceptable policy identifier. 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.

This part of ISO 15782 does not define any requirements for policy qualifiers.

8.2.7 Policy mappings

This non-critical extension allows a certificate issuer to indicate that one or more of that issuer's certificate policies is considered equivalent to another policy used in the subject CA's domain. The assignment of policy mappings is restricted to the CA, and may be further inhibited through the **policyConstraints** extension. This extension will support cross-certification by specifying the OIDs of equivalent policies.

CAs shall generate this extension for CA digital signature certificates where policy mapping is applicable and include a combination of **issuerDomainPolicy** field(s) and **subjectDomainPolicy** field(s) with the applicable **CertPolicyId** field(s).

Relying parties shall interpret the combination(s) of **issuerDomainPolicy** and **subjectDomainPolicy** certificate policy object identifiers as equivalent.

8.2.8 Subject alternative name

This extension provides a name that is bound to the subject’s certified public key. It may be implemented as either a critical or a non-critical extension. This extension should only be included when the CA chooses to stipulate the use of alternate names.

CAs which generate this extention shall be capable of populating the alternative name, **GeneralName,** with types

— **rfc822Name**,

— **ediPartyName**, and

— **uniformResourceIdentifier**.

The **subject** field of the certificate shall contain a **directoryName** that unambiguously identifies the subject.

An implementation that supports this extension is not required to be able to process all name forms, but relying parties shall process alternative names based on the naming constraints specified in 8.2.12.

NOTE If this extension field is present and is flagged critical, the subject field of the certificate could 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.

8.2.9 Issuer alternative name

This extension provides a name, in a form other than that of a distinguished name, for the certificate issuer. It may be implemented as either a critical or a non-critical extension.

CAs that implement this extension shall be capable of populating **GeneralName** with types

— **rfc822Name**,

— **ediPartyName**, and

— **uniformResourceIdentifier**.

An implementation that supports this extension is not required to be able to process all name forms, but relying parties shall process alternative names based on naming constraints specified in 8.2.12.

NOTE If this extension field is present and is flagged critical, the issuer field of the certificate or CRL could 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.

8.2.10 Subject directory attributes

This non-critical extension may convey any desired directory attribute values for the subject of the certificate. No attributes have been defined for this profile.

8.2.11 Basic constraints

This extension indicates whether the subject may act as a CA using the certified public key to sign certificates. If so, a certification path length constraint may also be specified. This extension shall be implemented as a critical extension.

CAs shall include this extension in all CA digital signature certificates. CAs shall indicate that this extension is critical by setting the criticality flag to “true”. CAs shall set the **cA** boolean flag to “True” for CA certificates, and enter a path length constraint if applicable. CAs may omit this extension for end entity certificates.

Relying parties shall verify that this extension is present in all CA digital signature certificates, and reject the processing of certificates issued by entities without the **cA** Boolean flag set to “True” (irrespective of the criticality of the extension). If this extension is not present, the certificate processing entity shall process the certificate as an end entity certificate. The certificate processing entity shall impose the **pathLenConstraint** field, if present.

8.2.12 Name constraints

This extension, for use only in CA certificates, indicates a name space within which all subject names in a subsequent certification path shall be located. This extension shall be implemented as a critical extension. Although this extension is not required in all CA certificates, it is recommended that name constraints be employed to the fullest possible extent.

CAs shall include this extension in applicable CA certificates and shall indicate that the extension is critical by setting the criticality flag to “true”.

a) The CA shall include the **PermittedSubtrees** and **excludedSubtrees** fields.

b) The CA shall be capable of populating **GeneralName** with types

- **rfc822Name**,
- **uniformResourceIdentifier**, and
- **directoryName**

and shall include the appropriate integer in the minimum and maximum fields of **GeneralSubtree** to indicate the name space as required.

c) Relying parties shall impose constraints for **GeneralNames** of types

- **rfc822Name**,
- **uniformResourceIdentifier**, and
- **directoryName**,

based on the following.

d) Restrictions for the

— **rfc822Name**,

— **dNSName**, and

— **uniformResourceIdentifier**

name forms are all expressed in terms of strings with wild card matching. ("*" is the wildcard character).

The minimum and maximum fields in general subtree are not used for these name forms. For **uniformResourceIdentifier** and **rfc822Name** names, the restriction applies to the host part of the name. Examples would be moo.bar.com; www*.bar.com; *.xyz.com. Restrictions of the form **directoryName** shall be applied to the subject field in the certificate and to the **subjectAltName** extensions of type **directoryName**.

8.2.13 Policy constraints

This extension specifies constraints which may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path. This extension shall be implemented as a critical extension.

CAs shall include this extension in applicable CA certificates, indicate that the extension is critical by setting the criticality flag to "true", and include the **requireExplicitPolicy** field with the applicable **SkipCerts** value. This indicates that in all certificates starting from the nominated CA (indicated by the **SkipCerts** value) in the certification path until the end of the certification path, it is necessary for the certificate to contain an acceptable policy identifier in the **certificatePolicies** extension. CAs shall also include the **inhibitPolicyMapping** field as applicable.

Relying parties shall use the **policyConstraints** information during the certification path validation process as specified in ISO/IEC 9594-8 | ITU-T Recommendation X.509 (1997).

8.2.14 CRL distribution points

This extension identifies the CRL distribution point or points to which a relying party should refer in order to ascertain if the certificate has been revoked. It may be implemented as either a critical or a non-critical extension. This extension provides a mechanism for assembling a CRL that contains certificates revoked for specific reason codes.

CAs may include this extension in CA and end entity certificates based on operational requirements and business risk. CAs shall indicate the criticality of this extension based on business risk.

a) If the extension is made critical, the CA shall ensure that the distribution points include certificates revoked with the reason codes **keyCompromise** or **cACompromise**, or both, whichever is applicable. If this extension is flagged critical, then a relying party shall not use the certificate without first retrieving and checking a CRL from one of the nominated distribution points including, at a minimum, the certificates revoked with the reason code **keyCompromise** (for an end entity certificate) or **cACompromise** (for a CA certificate).

b) If this extension is flagged critical and a relying party does not recognise the extension field type, then the system shall consider the certificate invalid.

c) If the extension is flagged non-critical and a relying party does not recognise the extension field type, then that system shall only use the certificate if it can acquire and check a complete CRL from the CA.

d) If this extension is used, a complete CRL is not required. If this extension is not used, the CA shall generate a complete CRL, but is not required to distribute it.

8.3 CRL extensions

8.3.1 Introduction

The following clauses describe the standard CRL extensions. The CRL extensions add information about the CRL and the CRL issuer, and provide mechanisms to control the size of the CRLs.

8.3.2 Authority key identifier

This non-critical CRL extension is described in 8.2.1.

8.3.3 Issuer alternative name

This CRL extension is described in 8.2.9.

8.3.4 CRL number

This non-critical CRL extension conveys a monotonically increasing sequence number for each CRL issued by a given CA through a given CA directory attribute or CRL distribution point directory attribute.

CAs shall include a monotonically increasing sequence number for each CRL issued by that CA (e.g. 1, 2, 3, ...). There are no requirements for implementations to perform any automated processing of this extension.

8.3.5 Issuing distribution point

This critical 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. This extension also indicates whether the CRL is an indirect CRL. An indirect CRL is a CRL that contains entries from CAs other than the authority that signed and issued the CRL.

CAs may generate this extension when required by business risks and practices. The use of indirect CRLs has timeliness, availability, access and risk implications. Indirect CRLs are not recommended for use in high-risk systems.

8.3.6 Delta CRL indicator

This critical CRL extension identifies a CRL as being a delta-CRL only. A relying party that does not understand this use of delta-CRLs should not use a CRL containing this extension, as the CRL may not be as complete as the user expects.

There are no requirements to support this extension.

8.4 CRL entry extensions

8.4.1 Introduction

The following clauses describe the standard CRL entry extensions. The CRL entry extensions add information about a specific entry within the CRL. An interrelationship exists between the certificate, CRL and CRL entry extension. Figure 4 illustrates this relationship.

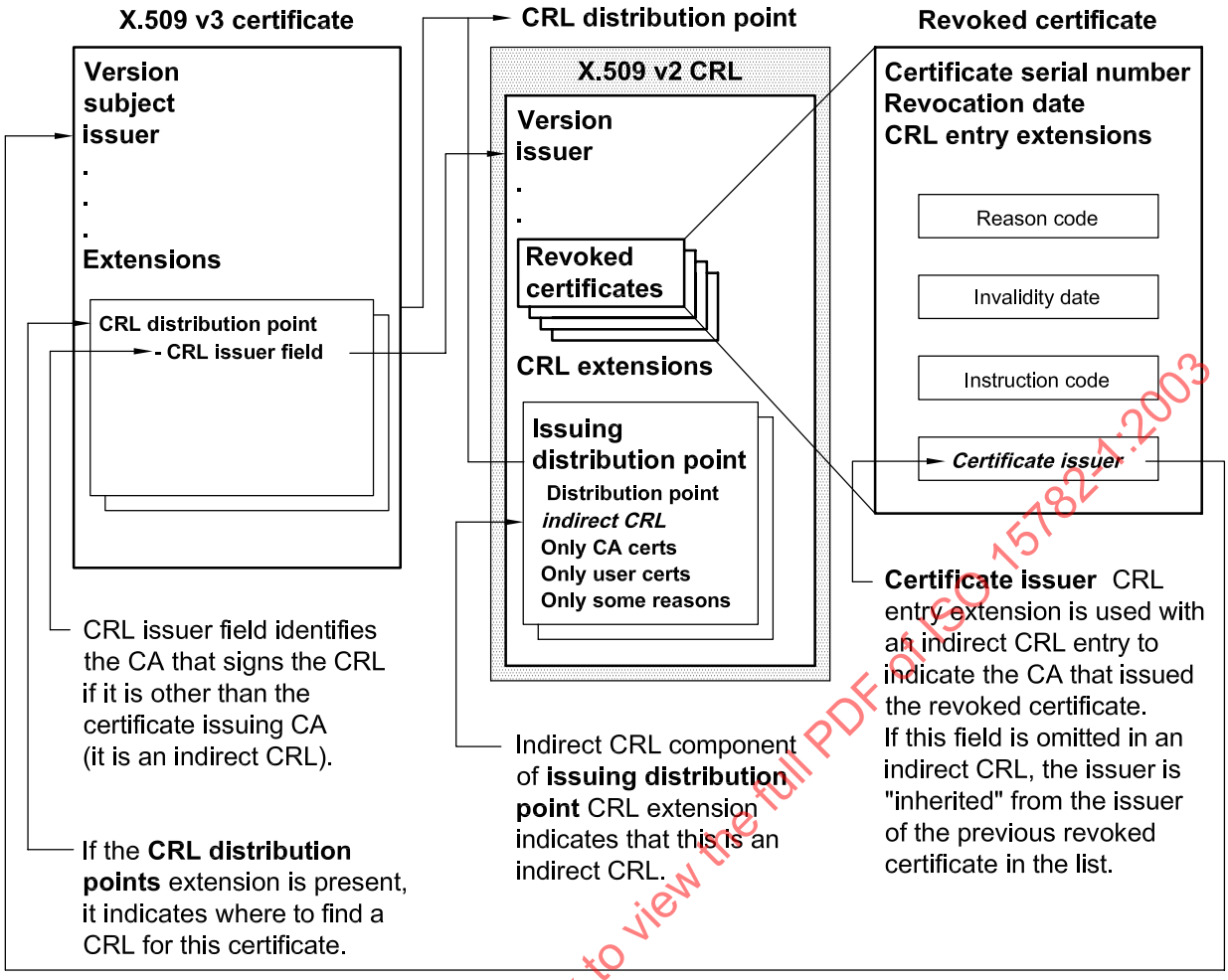


Figure 4 — Interrelationship of certificate, CRL and CRL entry extensions

8.4.2 Certificate issuer

This critical CRL entry extension identifies the certificate issuer associated with an entry in a CRL that has the **indirectCRL** indicator set. 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.

CAs shall generate this extension for indirect CRL entries, and shall include the **GeneralName** of the certificate issuer as it appears in the **issuer** field of the revoked certificate. Relying parties shall use this extension to associate the certificate issuer to a certificate entered on an indirect CRL.

8.4.3 Reason code

This non-critical CRL entry extension identifies the reason for the certificate revocation. The user can then decide, based on the reason for revocation, how much trust to place in the certificate.

CAs should populate this extension in all CRL entries, except that this extension should be absent if the reason code is **unspecified**. The reason code **removeFromCRL** is associated with Delta CRLs. Support for Delta CRLs is not required by this part of ISO 15782, and thus neither is support for **removeFromCRL**. There are no requirements for relying parties to perform any automated processing of this extension.

8.4.4 Hold instruction code

This non-critical CRL entry extension provides for inclusion of a registered instruction identifier to indicate the action to be taken after encountering a held certificate. There are no requirements for relying parties to perform any automated processing of this extension.

8.4.5 invalidityDate

This non-critical CRL entry extension should be used to identify the date at which the certificate should 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.

CAs should populate this extension in all CRL entries. There are no requirements for relying parties to perform any automated processing of this extension.

Annex A
(normative)

ASN.1 modules

A.1 General

The following ASN.1 module reflects the ASN.1 definitions in this part of ISO 15782[2]. These modules could be input to an ASN.1 compiler for the purpose of validating syntactic correctness.

A.2 Base certificate ASN.1

NOTE Several of the 1997 extensions are used. Many of the types are taken from X.501 and X.509 (1993), and could be **IMPORT**ed from the appropriate modules in those standards, if desired.

```
CertificateManagement {
        iso(1) member-body(2) us(840) x9-57(10040) module(1)
        certificateManagement(1) }

DEFINITIONS EXPLICIT TAGS ::= BEGIN

-- EXPORTS All;

IMPORTS
        authorityKeyIdentifier, basicConstraints, certificateIssuer,
        certificatePolicies, cRLDistributionPoints, cRLNumber,
        deltaCRLIndicator, extKeyUsage, holdInstructionCode,
        invalidityDate, issuerAltName, issuingDistributionPoint,
        keyUsage, nameConstraints, policyConstraints, policyMappings,
        privateKeyUsagePeriod, reasonCode, subjectAltName,
        subjectDirectoryAttributes, subjectKeyIdentifier

FROM CertificateExtensions
    Name, UniqueIdentifier

FROM SupportingDefinitions;
        --basic certificate definition

Certificate ::= SIGNED { EncodedCertificate }

EncodedCertificate ::= TYPE-IDENTIFIER.&Type( CertificateInfo )

CertificateInfo ::= SEQUENCE {
        Version         [0] Version DEFAULT v1,
        serialNumber        CertificateSerialNumber,
        signature       AlgorithmIdentifier{{SignatureAlgorithms}},
        issuer              Name,
        validity            Validity,
        subject             Name,
        subjectPublicKeyInfo    SubjectPublicKeyInfo,
```

2) This ASN.1 module is aligned with the version of [1] available at the time of publication of this part of ISO 15782.

```
        issuerUniqueID      [1] IMPLICIT UniqueIdentifier  OPTIONAL,
        subjectUniqueID     [2] IMPLICIT UniqueIdentifier  OPTIONAL,
        extensions          [3] Extensions { CertExtensions }  OPTIONAL
}

Version ::= INTEGER { v1(0), v2(1), v3(2) }

CertificateSerialNumber ::= INTEGER

AlgorithmIdentifier { ALGORITHM-ID:IOSet } ::= SEQUENCE {
        algorithm ALGORITHM-ID.&id({IOSet}),
        parameters    ALGORITHM-ID.&Type({IOSet}{@algorithm}) OPTIONAL
}

SignatureAlgorithms ALGORITHM-ID ::= {
        { OID id-rsa-signature              PARMS NULL }  |
        { OID id-rsa-signature-with-sha1    PARMS NULL },
        ...
}

Validity ::= SEQUENCE {
        notBeforeTime,
        notAfter   Time
}

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

SupportedAlgorithms ALGORITHM-ID ::= { ... }

Time ::= CHOICE {
        utcTime         UTCTime,
        generalizedTime   GeneralizedTime
}

Extensions { IOSet } ::= SEQUENCE OF Extension { IOSet }

Extension { IOSet } ::= SEQUENCE {
        extnID          EXTENSION.&id({IOSet}),
        critical        EXTENSION.&critical({IOSet}{@extnID})  DEFAULT FALSE,
        extnValue       OCTET STRING
        -- An "octet hole", the value of extnValue contains the DER
        -- encoding of a value of type &ExtnType, an open type, for
        -- the extension object identified by extnId.
}

EXTENSION ::= CLASS {
        &id         OBJECT IDENTIFIER  UNIQUE,
        &critical   BOOLEAN  DEFAULT FALSE,
        &ExtnType
}

         WITH SYNTAX
{ SYNTAX &ExtnType [CRITICAL &critical] IDENTIFIED BY &id }

CertExtensions EXTENSION ::= {
        authorityKeyIdentifier          |
```

```
        basicConstraints        |
        certificatePolicies     |
        cRLDistributionPoints   |
        extKeyUsage             |
        issuerAltName           |
        keyUsage                |
        nameConstraints         |
        policyConstraints       |
        policyMappings          |
        privateKeyUsagePeriod   |
        subjectAltName          |
        subjectDirectoryAttributes |
        subjectKeyIdentifier,
        ...  -- Expect other extension objects in future revisions of this part of ISO 15782--
}

CRLExtensions EXTENSION ::= {
        authorityKeyIdentifier  |
        cRLNumber               |
        deltaCRLIndicator       |
        issuerAltName           |
        issuingDistributionPoint,
        ...  -- Expect other extension objects in future revisions of this part of ISO 15782--
}

CRLEntryExtensions EXTENSION ::= {
        certificateIssuer   |
        holdInstructionCode |
        invalidityDate      |
        reasonCode,
        ...  -- Expect other extension objects in future revisions of this part of ISO 15782--
}

-- certificate request

CertReqDataSubmission ::= SIGNED { EncodedCertReqData }

EncodedCertReqData ::= TYPE-IDENTIFIER.&Type( CertReqData )

CertReqData ::= SEQUENCE {
        version    Version  DEFAULT v1,
        subject    Name,
        subjectPublicKeyInfo    SubjectPublicKeyInfo,
        requestedPeriod    [0] Validity  OPTIONAL,
        timeStamp       Time  OPTIONAL,
        extensions      [2] Extensions { CertExtensions } OPTIONAL
}

-- certificate renewal

CertRenDataSubmission ::= SIGNED { EncodedCertRenData }

EncodedCertRenData ::= TYPE-IDENTIFIER.&Type( CertRenData )

CertRenData ::= SEQUENCE {
        subject    Name,
        serialNumber  CertificateSerialNumber,
        validity        Validity
}
```

```
-- certificate revocation list (CRL)

CertificateList ::= CRL

CRL ::= SIGNED { EncodedCertificateList }

EncodedCertificateList ::= TYPE-IDENTIFIER.&Type(CertificateListInfo)

CertificateListInfo ::= SEQUENCE {
        version             Version OPTIONAL, -- if present, must be v2
        signature       AlgorithmIdentifier{{SignatureAlgorithms}},
        issuer              Name,
        thisUpdate          Time,
        nextUpdate          Time OPTIONAL,
        revokedCertificates     CRLEntryList  OPTIONAL,
        crlExtensions       [0] Extensions { CRLExtensions } OPTIONAL
}

CRLEntryList ::= SEQUENCE OF CRLEntry

CRLEntry ::= SEQUENCE{
        userCertificate         CertificateSerialNumber,
        revocationDate      Time,
        crlEntryExtensions Extensions { CRLEntryExtensions } OPTIONAL
}

-- certificate revocation

RevocationNotice ::= SIGNED { EncodedRevNoticeInfo }

EncodedRevNoticeInfo ::= TYPE-IDENTIFIER.&Type(RevocationNoticeInfo)

RevocationNoticeInfo ::= SEQUENCE {
        signature AlgorithmIdentifier{{SignatureAlgorithms}},
        issuer          Name,
        crlEntry    CRLEntry
}

-- information object classes --

ALGORITHM-ID ::= CLASS {
        &id   OBJECT IDENTIFIER  UNIQUE,
        &Type       OPTIONAL
}

            WITH SYNTAX { OID &id [PARMS &Type] }

-- parameterized types --

SIGNED { ToBeSigned } ::= SEQUENCE {
        toBeSigned    ToBeSigned,
        algorithm AlgorithmIdentifier{{SignatureAlgorithms}},
        signature BIT STRING
}

DSAPublicKey ::= INTEGER  -- public key y

DSAParameters ::= SEQUENCE {
        prime1    INTEGER,  -- modulus p
```

```
        prime2    INTEGER,  -- modulus q
        base INTEGER   -- base g
}

DSAMatchParameters ::= ModulusLength

ModulusLength ::= INTEGER  -- length of modulus in bits

RSAPublicKey ::= SEQUENCE {
        n      INTEGER,
        e      INTEGER
}

-- object identifier assignments

secsig OBJECT IDENTIFIER ::= {
        iso(1) identified-organization(3) oiw(14) secsig(3) }

secsigAlgorithm OBJECT IDENTIFIER ::= {
        secsig algorithm(2) }

id-rsa-signature OBJECT IDENTIFIER ::= {
        secsigAlgorithm 11 }

id-rsa-signature-with-sha1 OBJECT IDENTIFIER ::= {
        secsigAlgorithm 29 }

-- algorithm information objects

rsaSignature ALGORITHM-ID ::= {
        OID id-rsa-signature PARMS NULL }

rsaSignatureWithSHA1 ALGORITHM-ID ::= {
        OID id-rsa-signature-with-sha1 PARMS NULL }

END
```

A.3 Certificate extensions

```
CertificateExtensions {

        iso(1) member-body(2) us(840) x9-57(10040) module(1) certificateExtensions(2) }

        DEFINITIONS IMPLICIT TAGS ::= BEGIN

-- EXPORTS All; --

IMPORTS

        AlgorithmIdentifier {}, CertificateList, CertificateSerialNumber,
        EXTENSION, SupportedAlgorithms
                FROM CertificateManagement

        Attribute, DirectoryString {}, id-ce, Name, RelativeDistinguishedName, ub-name
                FROM SupportingDefinitions;
```

```
-- key and policy information extensions

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

AuthorityKeyIdentifier ::= SEQUENCE {
        keyIdentifier   [0] KeyIdentifier
}

KeyIdentifier ::= OCTET STRING

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

SubjectKeyIdentifier ::= KeyIdentifier

keyUsage EXTENSION ::= {
        SYNTAX          KeyUsage
        CRITICAL        TRUE
        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)
}

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

KeyPurposeId ::= OBJECT IDENTIFIER

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} )
```

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

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

PolicyInformation ::= SEQUENCE {
        policyIdentifier    CertPolicyId,
        policyQualifiers    PolicyQualifiers  OPTIONAL
}

PolicyQualifiers ::=  SEQUENCE SIZE(1..MAX) OF PolicyQualifierInfo

CertPolicyId ::= OBJECT IDENTIFIER

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

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

CERT-POLICY-QUALIFIER ::= CLASS {
        &id         OBJECT IDENTIFIER  UNIQUE,
        &QualifierOPTIONAL
}

WITH SYNTAX

        { POLICY-QUALIFIER-ID &id [QUALIFIER-TYPE &Qualifier] }

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

PolicyMappingsSyntax ::= SEQUENCE SIZE(1..MAX) OF PolicyMapping

PolicyMapping ::= SEQUENCE {
        issuerDomainPolicy    CertPolicyId,
        subjectDomainPolicy   CertPolicyId
}

-- certificate subject and issuer attributes extensions

subjectAltName EXTENSION ::= {
        SYNTAX      GeneralNames
        --   CRITICAL          TRUE --
        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] EXPLICIT Name,
        ediPartyName            [5] EDIPartyName,
        uniformResourceIdentifier   [6] IA5String,
        iPAddress               [7] OCTET STRING,
        registeredID            [8] OBJECT IDENTIFIER
        --
        -- Other choices defined in X.509 are not used in this part of ISO 15782 --
}

OTHER-NAME ::= TYPE-IDENTIFIER

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

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

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

AttributesSyntax ::= SEQUENCE SIZE (1..MAX) OF Attribute
        -- certification path constraints extensions

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

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

nameConstraints EXTENSION ::= {
        SYNTAX      NameConstraintsSyntax
        -- CRITICAL      TRUE --
        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)

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

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

SkipCerts ::= INTEGER (0..MAX)

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

-- basic CRL extensions

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

CRLNumber ::= INTEGER (0..MAX)

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)
}

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

HoldInstruction ::= OBJECT IDENTIFIER

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

```
-- CRL distribution points and delta-CRL extensions

cRLDistributionPoints EXTENSION ::= {
        SYNTAX      CRLDistPointsSyntax
        -- CRITICAL      TRUE --
        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)
}

issuingDistributionPoint EXTENSION ::= {
        SYNTAX      IssuingDistPointSyntax
        CRITICAL            TRUE
        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
}

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

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

BaseCRLNumber ::= CRLNumber
```

```
-- object identifier assignments --

id-ce-subjectDirectoryAttributes OBJECT IDENTIFIER ::= { id-ce   9 }
id-ce-subjectKeyIdentifier       OBJECT IDENTIFIER ::= { id-ce 14 }
id-ce-keyUsage                   OBJECT IDENTIFIER ::= { id-ce 15 }
id-ce-privateKeyUsagePeriod      OBJECT IDENTIFIER ::= { id-ce 16 }
id-ce-subjectAltName             OBJECT IDENTIFIER ::= { id-ce 17 }
id-ce-issuerAltName              OBJECT IDENTIFIER ::= { id-ce 18 }
id-ce-basicConstraints           OBJECT IDENTIFIER ::= { id-ce 19 }
id-ce-cRLNumber                  OBJECT IDENTIFIER ::= { id-ce 20 }
id-ce-reasonCode                 OBJECT IDENTIFIER ::= { id-ce 21 }
id-ce-instructionCode            OBJECT IDENTIFIER ::= { id-ce 23 }
id-ce-invalidityDate             OBJECT IDENTIFIER ::= { id-ce 24 }
id-ce-deltaCRLIndicator          OBJECT IDENTIFIER ::= { id-ce 27 }
id-ce-issuingDistributionPoint   OBJECT IDENTIFIER ::= { id-ce 28 }
id-ce-certificateIssuer          OBJECT IDENTIFIER ::= { id-ce 29 }
id-ce-nameConstraints            OBJECT IDENTIFIER ::= { id-ce 30 }
id-ce-cRLDistributionPoints      OBJECT IDENTIFIER ::= { id-ce 31 }
id-ce-certificatePolicies        OBJECT IDENTIFIER ::= { id-ce 32 }
id-ce-policyMappings             OBJECT IDENTIFIER ::= { id-ce 33 }
-- deprecated                    OBJECT IDENTIFIER ::= { id-ce 34 }
id-ce-authorityKeyIdentifier     OBJECT IDENTIFIER ::= { id-ce 35 }
id-ce-policyConstraints          OBJECT IDENTIFIER ::= { id-ce 36 }
id-ce-extKeyUsage                OBJECT IDENTIFIER ::= { id-ce 37 }
holdInstruction                  OBJECT IDENTIFIER ::= {

iso(1) member-body(2) us(840) x9-57(10040) hi(2) }
none            OBJECT IDENTIFIER ::= { holdInstruction none(1) }
callIssuer      OBJECT IDENTIFIER ::= { holdInstruction callIssuer(2) }
reject          OBJECT IDENTIFIER ::= { holdInstruction reject(3) }
pickupToken     OBJECT IDENTIFIER ::= { holdInstruction pickupToken(4) }
END
```

A.4 Supporting definitions

```
SupportingDefinitions {
        iso(1) member-body(2) us(840) x9-57(10040) module(1)  supportingDefinitions(3) }

DEFINITIONS EXPLICIT TAGS ::= BEGIN
--
-- This module provides a collection of ASN.1 notation needed to support the
-- management of public key certificates in a financial services environment.
--
-- EXPORTS All;
-- IMPORTS None;
--
-- Defined in X.520|ISO/IEC 9594-6 module SelectedAttributeTypes
--

DirectoryString { INTEGER:maxSize } ::= CHOICE {
        teletexString       TeletexString (SIZE(1..maxSize)),
        printableString         PrintableString (SIZE(1..maxSize)),
        universalString     UniversalString (SIZE(1..maxSize)),
        bmpString           BMPString (SIZE(1..maxSize)),
        utf8String      UTF8String (SIZE(1..maxSize))
}
```

```
UniqueIdentifier ::= BIT STRING
        --
        -- Defined in X.520|ISO/IEC 9594-6 module UpperBounds
        --

ub-name INTEGER ::= 32768
        --
        -- Defined in X.501|ISO/IEC 9594-2 module InformationFramework
        --

Name ::= CHOICE {  -- only one possibility for now --
        rdnSequence  RDNSequence
}

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::= SET SIZE(1..MAX) OF AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
        type  ATTRIBUTE.&id({SupportedAttributes}),
        valueATTRIBUTE.&Type({SupportedAttributes}{@type})
}

Attribute ::= SEQUENCE {
        type  ATTRIBUTE.&id({SupportedAttributes}),
        values     SET SIZE(1..MAX) OF
         ATTRIBUTE.&Type({SupportedAttributes}{@type})
}

SupportedAttributes ATTRIBUTE ::= { ... }
        --
        -- This simplified version of the ATTRIBUTE class defined in X.501|ISO/IEC 9594-2
        -- module InformationFramework produces syntax conformant with that Directory
        -- standard, but only includes those fields needed for public key certificates.
        --

ATTRIBUTE ::= CLASS {
        &Type,
        &id   OBJECT IDENTIFIER  UNIQUE
}

        WITH SYNTAX { WITH SYNTAX &Type ID &id }
        --
        -- Defined in X.501|ISO/IEC 9594-2 module UsefulDefinitions
        --

id-ce OBJECT IDENTIFIER ::= {  joint-iso-ccitt ds(5) 29 }  -- Certificate Extension


END
```

Annex B
(normative)

Parameters and parameter inheritance[3)]

B.1 Public key parameters

The DSA and ECDSA digital signature algorithms require that parameters be used to generate and validate digital signatures. RSA does not require parameters. The ASN.1 type **Algorithmidentifier**, which may include a **parameters** field, can occur in three fields in a certificate:

a) the **signature** field, contained in the signed envelope of the certificate, which identifies the algorithm used to sign the certificate and may optionally contain parameters;

b) the **subjectPublicKeyInfo** field, contained in the signed envelope of the certificate, which specifies the certificate subject's public key and includes an **algorithm** field that contains any parameters applicable to that subject public key; and

c) the **SIGNED** parameterized type, which includes an **algorithm** field as well as the encrypted message digest. This **Algorithmidentifier** field is not within the signed envelope of the certificate.

B.2 Public key parameter processing requirements

Before any certificate or certification path can be validated, the system performing the validation shall obtain a public key and any associated parameters required by the signature algorithm in an authenticated manner. This key and these associated parameters are therefore trusted. The parameters (as well as the public key) used to validate the first certificate in the certification path shall be the initial trusted parameters. At each subsequent certificate validation along the certification path, the parameters used shall be taken from the **subjectPublicKeyInfo** field of the previously validated CA certificate containing the public key corresponding to the private key used to sign the certificate being validated, if parameters are included in the **subjectPublicKeyInfo** field of the previously validated certificate.

However, the **parameters** field of the **subjectPublicKeyInfo** field of a certificate in the certification path may be null for algorithms that require parameters. In this case:

a) the **algorithm** field of the **subjectPublicKeyInfo** field of the previous certificate in the certification path (or, in the case of the first certificate in the path, the algorithm of the initial trusted public key) shall be identical to the **algorithm** field of the **subjectPublicKeyInfo** field of the certificate being validated, or the validation fails;

b) the parameters used to validate the previous certificate in the path (or, in the case of the first certificate in the path, the initial trusted parameters) shall be used to validate the current certificate in the path.

CAs that sign certificates, including subject public keys for an algorithm that requires parameters, shall ensure that either

— valid parameters for the certified public key are stated in the certified **subjectPublicKeyInfo** field, or,

— if parameters are not included in the certified **subjectPublicKeyInfo** field, the algorithm stated the certified **subjectPublicKeyInfo** field is the algorithm used to sign the certificate, and the parameters used to sign the certificate are valid for the certified public key.

3) For discreet log-based algorithms such as Diffie-Hellman, DSA and ECDSA.

B.3 Examples

B.3.1 Example 1

Consider a certification path of three CAs and an End Entity (CA_4, CA_5, CA_6 and E_1):

— CA_4 is a trusted CA and the entity validating the certification path has obtained the DSA public key and parameters for CA_4 in an authenticated manner;

— CA_4 has issued the certificate CA_{4dsa}<< CA_{5dsa} >>, i.e. CA_4 has used the DSA to sign a certificate for CA_5 that certifies a DSA public key;

— CA_5 has issued the certificate CA_{5dsa}<< CA_{6dsa} >>, i.e. CA_5 has used the DSA to sign a certificate for CA_6 that certifies a DSA public key.

— CA_6 has issued the certificate CA_{6dsa}<< E_{1dsa} >>, i.e. CA_6 has used the DSA to sign a certificate for E_1 that certifies a DSA public key.

The certificates CA_{4dsa}<< CA_{5dsa} >>, $CA5_{5dsa}$<< CA_{6dsa} >>, CA_{6dsa}<< E_{1dsa} >> certify DSA keys without certifying parameters. The parameters that apply to these three certificates are then the authenticated parameters for CA_4.

B.3.2 Example 2

Consider a certification path of three CAs (CA_1, CA_2, and CA_3):

— CA_1 is a trusted CA and the entity validating the certification path has obtained the DSA public key and parameters for CA_1 in an authenticated manner;

— CA_1 has issued the certificate CA_{1dsa}<< CA_{2rsa} >>, i.e. CA_1 has used the DSA to sign a certificate for CA_2 that certifies an RSA public key;

— CA_2 has issued the certificate $CA2_{2rsa}$<< CA_{3dsa} >>, i.e. CA_2 has used the RSA to sign a certificate for CA_3 that certifies a DSA public key.

If the certificate $CA2_{2rsa}$<< CA_{3dsa} >> does not state DSA parameters in its **subjectPublicKeyInfo** field, then the certification path is invalid, and the certificate $CA2_{2rsa}$<< CA_{3dsa} >> cannot be used to validate any signatures.

Annex C
(normative)

Financial institution profile for Version 3 certificate extensions

C.1 Introduction

The profile described in the following clauses provides a recommended implementation for Version 3 X.509 certificates and Version 2 certificate revocation lists (CRLs) for the financial community. Because of the generalities inherent in the X.509 standard, this profile is recommended for enhancing interoperability as well as communications security. The profile provides a recommended implementation for each extension and field of the X.509 certificate and CRL.

C.2 Certificate profile tables

C.2.1 General

This clause lists the protocol elements that CAs shall generate in order to add extensions to the certificate as well as the protocol elements that relying parties shall understand if the extension is to be correctly processed. The lists are presented in tabular format with five major column headings. See Tables C.1 to C.10.

The *Item* and *Table* columns are provided for cross-referencing. The numbers in the Item column are the row numbers. The numbers in the Table column indicate the table number (followed by a “1”) and an Item number. These two columns are used together to point to subelements.

The *Protocol Element* column refers to the name of the ASN.1 field taken from Recommendation X.509.

The *Proc* column indicates if processing of the element is mandatory or optional for compliance with this part of ISO 15782.

The *Generation* column specifies the level of support required for each element. The level of support contains the support classifications required by compliant relying parties that process certificates. The Generation column is divided into two certificate types: *Signature* and *KM* (for key management). The Signature column is further divided into *CA* and *End Entity* certificates (no differentiation is needed for KM certificates). The information that is to appear in each type of certificate is identified within each respective column. The *Notes* column refers to additional information supplied at the end of the table.

C.2.2 Support classification

Each of the protocol elements listed in C.2.3 and C.2.4 is designated as having a support requirement of mandatory or optional. Where protocol elements are nested (i.e. the elements contain subelements), the requirement to support the nested element is relevant only when the immediately containing (parent) element is supported (e.g. if the top level element is designated as optional (o), its subelements may still be designated as mandatory (m) for the reason that, if the top level element is implemented, the subelements designated as mandatory shall also be implemented).

C.2.3 Static capability

The following classifications are used to specify static conformance (i.e. capability).

mandatory support (m)
CAs shall be able to generate the protocol element. Relying parties shall be able to receive the protocol element and perform all associated procedures (i.e. implying the ability to handle both the syntax and the semantics of the element) as relevant. Populating the information of this protocol element is an implementation detail based on compliance with this part of ISO 15782.

optional (o)
CAs are not required to support the generation of the protocol element. If support is claimed (i.e. if the optional element is implemented), the element shall be treated as if it were specified as mandatory support, and the sub-elements, if present, shall be supported as specified (i.e. an optional element may have subelements indicated as mandatory, "m"; this indicates that if the optional element is implemented, the subelements shall also be implemented as specified). Implementations claiming to perform the processing of certificates may ignore the protocol element and continue processing the certificate, unless it is flagged "critical". See Recommendation X.509 for the rules for processing critical extensions.

not applicable (-)
The element is not applicable in the particular context in which the classification is used.

C.2.4 Dynamic behaviour

The following classifications are used to specify dynamic conformance (i.e. behaviour).

prohibited (x)
CAs shall ensure that the element is never generated. Implementations performing certificate processing shall generate and return an appropriate error if a prohibited element is encountered.

critical (k)
The element, if present in the certificate and not recognized by the relying party, shall cause the system to consider the certificate invalid. The element, if present in a CRL entry and not recognized by the relying party, shall indicate to the user that the CRL may not be as complete as the user expects.

required (r)
the information for this protocol element shall be populated upon certificate generation.

Table C.1 — Base certificate

Item	Protocol element	Proc	Generation			Notes	Table
			Signature		KM		
			CA	End entity			
1.	Certificate	m	mr	mr	mr		
2.	Version	m	mr	mr	mr		
3.	SerialNumber	m	mr	mr	mr		
4.	Signature	m	mr	mr	mr		C2/1
5.	Issuer	m	mr	mr	mr		
6.	Validity	m	mr	mr	mr		
7.	notBefore	m	mr	mr	mr		
8.	notAfter	m	mr	mr	mr		
9.	Subject	m	mr	mr	mr		
10.	SubjectPublicKeyInfo	m	mr	mr	mr		
11.	algorithm	m	mr	mr	mr		C2/1
12.	subjectPublicKey	m	mr	mr	mr		
13.	IssuerUniqueIdentifier	x	x	x	x		
14.	SubjectUniqueIdentifier	x	x	x	x		
15.	Extension	m	mr	mr	mr		C3/1

Table C.2 — Algorithm identifier

Item	Protocol element	Proc	Generation			Notes	Table
			Signature		KM		
			CA	End entity			
1.	AlgorithmIdentifier						
2.	algorithm	m	mr	mr	mr		
3.	parameters	m	o	o	o	1	
NOTE See Annex E for the encoding of public keys and associated parameters.							

Table C.3 — Extensions

Item	Protocol element	Proc	Generation			Notes	Table
			Signature		KM		
			CA	End entity			
1.	Extensions	m	mr	mr	mr		
2.	Extension	m	mr	mr	mr		
3.	extnID	m	mr	mr	mr		
4.	critical	m	mr	mr	mr		
5.	extnValue	m	mr	mr	mr		

Table C.4 — Standard extensions

Item	Protocol element	Proc	Generation			Notes	Table
			Signature		KM		
			CA	End entity			
1.	AuthorityKeyIdentifier	o	m	m	m	1	C5/1
2.	SubjectKeyIdentifier	o	m	m	m	1	C5/15
3.	KeyUsage	m	kmr	kmr	kmr		C5/16
4.	ExtendedKeyUsage	o	o	o	o		C5/26
5.	PrivateKeyUsagePeriod	o	o	o	—		C5/27
6.	CertificatePolicies	m	(k)mr	(k)mr	(k)mr		C5/30
7.	PolicyMappings	m	m	—	—		C5/36
8.	SubjectAltName	m	(k)m	(k)m	(k)m		C5/39
9.	IssuerAltName	m	(k)m	(k)m	(k)m		C5/39
10.	SubjectDirectoryAttributes	o	o	o	o		—
11.	BasicConstraints	m	kmr	kmr	o		C5/51
12.	NameConstraints	m	km	—	—	2	C5/54
13.	PolicyConstraints	m	km	—	—		C5/71
14.	CRLDistributionPoints	o	o	o	o		C5/74
NOTE 1 Though not mandatory, this extension is recommended for certificate processing.							
NOTE 2 Population of this extension is encouraged to the fullest extent possible.							

Table C.5 — Standard extension syntax

Item	Protocol element	Proc	Generation			Notes	Table
			Signature		KM		
			CA	End entity			
1.	AuthorityKeyIdentifier						
2.	KeyIdentifier	m	m	m	m	7	
3.	AuthorityCertIssuer	o	o	o	o		
4.	GeneralName						
5.	otherName	o	o	o	o		
6.	rfc822Name	o	o	o	o		
7.	dNSName	o	o	o	o		
8.	x400Address	o	o	o	o		
9.	directoryName	o	o	o	o		
10.	ediPartyName	o	o	o	o		
11.	uniformResourceIdentifier	o	o	o	o		
12.	iPAddress	o	o	o	o		
13.	registeredID	o	o	o	o		
14.	AuthorityCertSerialNumber	o	o	o	o		
15.	SubjectKeyIdentifier	m	m	m	m		
16.	KeyUsage						
17.	digitalSignature	m	m	m	—		
18.	nonRepudiation	o	o	o	o		
19.	keyEncipherment	m	—	—	m		
20.	dataEncipherment	m	—	—	m		
21.	keyAgreement	m	—	—	m		
22.	keyCertSign	m	m	—	—		
23.	cRLSign	m	m	—	—		
24.	encipherOnly	o	—	—	o		
25.	decipherOnly	o	—	—	o		
26.	KeyPurposeId	m	m	m	m		
27.	PrivateKeyUsagePeriod						
28.	NotBefore	m	m	m	m	5	
29.	NotAfter	m	m	m	m	5	
30.	PolicyInformation						
31.	PolicyIdentifier	m	mr	mr	mr	1	
32.	PolicyQualifiers	o	o	o	o		
33.	PolicyQualifierInfo						
34.	PolicyQualifierId	m	m	m	m	6	
35.	Qualifier	o	o	o	o		
36.	PolicyMappingsSyntax						
37.	IssuerDomainPolicy	m	mr	—	—		
38.	SubjectDomainPolicy	m	mr	—	—		

Table C.5 (*continued*)

Item	Protocol element	Proc	Generation			Notes	Table
			Signature		KM		
			CA	End entity			
39.	GeneralName						
40.	otherName	o	o	o	o		
41.	rfc822Name	o	o	o	o		
42.	dNSName	o	o	o	o		
43.	x400Address	o	o	o	o		
44.	directoryName	o	o	o	o		
45.	ediPartyName	o	m	m	m		
46.	nameAssigner	o	m	m	m		
47.	partyName	o	m	m	m	4	
48.	uniformResourceIdentifier	m	m	m	m		
49.	iPAddress	o	o	o	o		
50.	registeredID	m	m	m	m		
51.	BasicConstraintsSyntax						
52.	cA	m	mr	o	—	d(false)	
53.	pathLenConstraint	m	m	—	—		
54.	NameConstraintsSyntax						
55.	PermittedSubtrees	m	mr	—	—	-	
56.	GeneralSubtree						
57.	Base	m	mr	—	—	3	
58.	GeneralName						
59.	otherName	o	o	—	—		
60.	rfc822Name	m	m	—	—		
61.	dNSName	o	o	—	—		
62.	x400Address	o	o	—	—		
63.	directoryName	m	m	—	—		
64.	ediPartyName	o	o	—	—		
65.	uniformResourceIdentifier	m	m	—	—		
66.	iPAddress	o	o	—	—		
67.	registeredID	o	o	—	—		
68.	Minimum	o	o	—	—	d(0), 2	
69.	Maximum	o	o	—	—		
70.	ExcludedSubtrees	m	m	—	—		C5/55
71.	PolicyConstraintsSyntax						
72.	RequireExplicitPolicy	m	m	—	m		
73.	InhibitPolicyMapping	m	m	—	m		
74.	CRLDistPointsSyntax						
75.	DistributionPoint	o	o	o	o		
76.	DistributionPointName	o	o	o	o		
77.	FullName	o	o	o	o		

Table C.5 (*continued*)

Item	Protocol element	Proc	Generation			Notes	Table
			Signature		KM		
			CA	End entity			
78.	GeneralName						
79.	otherName	o	o	o	o		
80.	rfc822Name	o	o	o	o		
81.	dNSName	o	o	o	o		
82.	x400Address	o	o	o	o		
83.	directoryName	o	o	o	o		
84.	ediPartyName	o	o	o	o		
85.	uniformResourceIdentifier	o	o	o	o		
86.	iPAddress	o	o	o	o		
87.	nameRelativeToCRLIssuer	o	o	o	o		
88.	Reasons						
89.	ReasonFlags						
90.	unused	o	o	o	o		
91.	keyCompromise	m	m	m	m		
92.	caCompromise	m	m	m	m		
93.	affiliationChanged	m	m	m	m		
94.	superseded	m	m	m	m		
95.	cessationOfOperation	m	m	m	m		
96.	certificateHold	m	m	m	m		
97.	CRLIssuer	m	mr	mr	mr		
98.	GeneralName						
99.	otherName	o	o	o	o		
100.	rfc822Name	o	o	o	o		
101.	dNSName	o	o	o	o		
102.	x400Address	o	o	o	o		
103.	directoryName	m	m	m	m		
104.	ediPartyName	o	o	o	o		
105.	uniformResourceIdentifier	o	o	o	o		
106.	iPAddress	o	o	o	o		
107.	registeredID	o	o	o	o		

If the **requireExplicitPolicy** field is present in the **policyConstraints** extension, this field shall include at least one of the policies applicable to the certificate.

The *minimum* attribute shall always be present if the extension is included in the certificate.

Although the **nameConstraints** extension is not always required to be present in a certificate, the *base* attribute shall always be present if **nameConstraints** is present.

partyName shall always be present if **ediPartyname** is included in the certificate.

One or both of the **notBefore** and **notAfter** elements shall be present in this extension.

PolicyQualifierId shall be present if **policyQualifierInfo** is included in the certificate.

If the **AuthorityKeyIdentifier** is present, **then keyIdentifier** shall be present.

Table C.6 — CRL

Item	Protocol eElement	Process	Generate	Notes	Table
1.	CertificateList				
2.	version	m	mr		
3.	Signature	m	mr		C2/2
4.	Issuer	m	mr		
5.	ThisUpdate	m	mr		
6.	nextUpdate	m	mr		
7.	RevokedCertificates	m	mr		
8.	userCertificate	m	mr		
9.	revocationDate	m	mr		
10.	crlEntryExtensions	m	mr		C9
11.	CrlExtensions	m	mr		C8

Table C.7 — CRL extensions

Item	Protocol element	Process	Generate	Notes	Table
1.	AuthorityKeyIdentifier	o	m		C5/1
2.	IssuerAltName	o	m		C5/39
3.	CRLNumber	o	mr		C8/1
4.	IssuingDistributionPoint	o	o		C8/2
5.	DeltaCRLIndicator	o	o		C8/8

Table C.8 — CRL extension syntax

Item	Protocol element	Process	Generate	Notes	Table
1.	CRLNumber	m	mr		
2.	IssuingDistPointSyntax	m	m		
3.	DistributionPoint	o	o		C5/75
4.	OnlyContainsUserCerts	m	m	d(false)	
5.	OnlyContainsCACerts	m	m	d(false)	
6.	OnlySomeReasons	o	o		C5/89
7.	IndirectCRL	m	mr	d(false)	
8.	BaseCRLNumber	m	m	1	
The value of this element shall be identical to the value in the CRLNumber extension of the base certificate.					

Table C.9 — CRL entry extensions

Item	Protocol element	Process	Generate	Notes	Table
1.	ReasonCode	o	m		C10/1
2.	HoldInstructionCode	o	o		
3.	InvalidityDate	o	m		
4.	CertificateIssuer	m	km		

Table C.10 — CRL entry extension syntax

Item	Protocol element	Process	Generate	Notes	Table
1.	CRLReason				
2.	unspecified	m	m		
3.	keyCompromise	m	m		
4.	cACompromise	m	m		
5.	affiliationChanged	m	m		
6.	superseded	m	m		
7.	cessationOfOperation	m	m		
8.	certificateHold	m	m		
9.	removeFromCRL	o	o		

Annex D
(normative)

Object identifiers and attributes

D.1 General

This annex lists all of the object identifiers used in this part of ISO 15782. Where the object identifier is used to identify some information object, such as an algorithm, the appropriate specification is also given. These specifications use the class definitions and syntax of X.500. The following definitions are used below.

ID ::= OBJECT IDENTIFIER

secsig ID ::= { iso(1) identified-organization(3) oiw(14) secsig(3) }

x9cm ID ::= { iso (1) member-body (2) us(840) x9-30-3(10040) }

module ID ::= {x9cm 1}

heldinstruction ID ::= {x9cm 2}

attribute ID ::= {x9cm 3}

D.2 Algorithms

Object Identifiers (OIDs) required for the implementation of this part of ISO 15782 are defined under the TC 68 arc and shall be taken from the OIDs published on the TC 68 web site.

D.3 Modules

This clause lists the object identifier for the ASN.1 module defined in Annex A.

id-x9f1-cert-mgmt ::= { module 1 }

D.4 Certificate Hold Instructions

id-holdinstruction-none ID ::= {holdinstruction 1}

id-holdinstruction-callIssuer ID ::= {holdinstruction 2}

id-holdinstruction-reject ID ::= {holdinstruction 3}

id-holdinstruction-pickupToken ID ::= {holdinstruction 4}

Annex E
(normative)

Encoding of public keys and associated parameters

E.1 DSA public keys

The DSA public key shall be encoded as an **INTEGER**; this encoding shall be used as the contents octets of the **subjectPublicKey** component (a **BIT STRING**) of the **SubjectPublicKeyInfo** data element.

```
DSAPublicKey ::= INTEGER     -- public key y

DSAParameters ::= SEQUENCE {
        prime1    INTEGER,    -- modulus p
        prime2    INTEGER,    -- modulus q
        base INTEGER }    -- base g

DSAMatchParameters ::= ModulusLength

ModulusLength ::= INTEGER    -- length of modulus in bits
```

For the DSA, the contents octets of the signature **BIT STRING** shall be interpreted as being the DER encoding of the type

```
SEQUENCE {

        rINTEGER,
        s      INTEGER }
```

That is, the encoding of the sequence is wrapped inside a **BIT STRING**.

Both **dsa-with-sha1** and **sha1** have **NULL** parameters. Implementations should therefore always insert the **NULL** parameter in the **sha1** algorithm identifier, and in the **dsa-with-sha1** algorithm identifier.

NOTE 1 In the past, some implementations have omitted the parameters.

NOTE 2 According to the definition above, the DSA is applied to the DER encoding of the hash, which has a length of 22 bytes (tag, length, and contents). Since the input to the DSA signature process is only 160 bits, the tag and length bytes of the DER encoding are not included in the computation of the signature; only the actual contents of the encoding (the hash) is signed.

The **SIGNED** type does not include the signature algorithm identifier in the signature computation. In order to deter attacks based upon modifying the hash or signature algorithm identifiers conveyed in the message, applications using the **SIGNED** type should include the algorithm identifier in the actual data being signed.

E.2 ECDSA public keys

E.2.1 General

This clause provides the syntax for elliptic curve parameters and keys according to abstract syntax notation one (**ASN.1**). While it is not required that elliptic curve parameters and keys be represented with **ASN.1** syntax, if they are so represented, then their syntax shall be as defined here. These **ASN.1** definitions shall be encoded using distinguished encoding rules (DER).

The object identifier **ansi-X9.62** represents the root of the tree containing all object identifiers defined in this part of ISO 15782, and has the following value:

ansi-X9-62 OBJECT IDENTIFIER ::= { iso(1) member-body(2) us(840) 10045 }

E.2.2 Syntax for finite field identification

The following provides the abstract syntax definitions for the finite fields defined in this part of ISO 15782.

A finite field shall be identified by a value of type **FieldID**:

```
FieldID { FIELD-ID:IOSet } ::= SEQUENCE {
        fieldType FIELD-ID.&id({IOSet}),
        parameters    FIELD-ID.&Type({IOSet}{@fieldType})
}

FieldTypes FIELD-ID ::= {
        { Prime-p       IDENTIFIED BY prime-field                } |
        { Characteristic-two      IDENTIFIED BY characteristic-two-field  },
        ...
}

FIELD-ID ::= TYPE-IDENTIFIER  -- ISO/IEC 8824-2:1998(E), Annex A
```

NOTE **FieldID** is a parameterized type composed of two components, **fieldType** and parameters. These components are specified by the fields `&id` and `&Type`, which form a template for defining sets of information objects, instances of the class **FIELD-ID**. This class is based on the useful information object class **TYPE-IDENTIFIER**, specified in ISO/IEC 8824-2:—[4] | ITU-T Recommendation X.681:1997, Annex A. In an instance of **FieldID**, "**fieldType**" will contain an object identifier value that uniquely identifies the type contained in "`parameters`". The effect of referencing "**fieldType**" in both components of the **fieldID** sequence is to tightly bind the object identifier and its type.

The information object set **FieldTypes** is used as the single parameter in a reference to type **FieldID**. **FieldTypes** contains two objects followed by the extension marker ("..."). Each object, which represents a finite field, contains a unique object identifier and its associated type. The values of these objects define all of the valid values that may appear in an instance of **fieldID**. The extension marker allows backward compatibility with future versions of this part of ISO 15782 which may define objects to represent additional kinds of finite fields.

The object identifier **id-fieldType** represents the root of a tree containing the object identifiers of each field type. It has the following value:

id-fieldType OBJECT IDENTIFIER ::= { ansi-X9-62 fieldType(1) }

The object identifiers **prime-field** and **characteristic-two-field** name the two kinds of fields defined in this part of ISO 15782. They have the following values:

prime-field OBJECT IDENTIFIER ::= { id-fieldType 1 }

characteristic-two-field OBJECT IDENTIFIER ::= { id-fieldType 2 }

Prime-p ::= INTEGER **-- Finite field F(p), where p is an odd prime**

```
Characteristic-two ::= SEQUENCE {
        m           INTEGER,    -- Field size 2^m
        basis       CHARACTERISTIC-TWO.&id({BasisTypes}),
        parameters  CHARACTERISTIC-TWO.&Type({BasisTypes}{@basis})
}
```

4) To be published. (Revision of ISO 8824-2:1998)

```
BasisTypes CHARACTERISTIC-TWO::= {
        { NULL          IDENTIFIED BY gnBasis } |
        { Trinomial     IDENTIFIED BY tpBasis  } |
        { Pentanomial IDENTIFIED BY ppBasis },
        ...
}

--
-- Trinomial basis representation of F2^m
-- Integer k for reduction polynomial xm + xk + 1
--

Trinomial ::= INTEGER

Pentanomial ::= SEQUENCE {
        --
        -- Pentanomial basis representation of F2^m
        -- reduction polynomial integers k1, k2, k3
        -- f(x) = x**m + x**k3 + x**k2 + x**k1 + 1
        --
        k1    INTEGER,
        k2    INTEGER,
        k3    INTEGER
}

CHARACTERISTIC-TWO ::= TYPE-IDENTIFIER
```

The object identifier **id-characteristic-two-basis** represents the root of a tree containing the object identifiers for each type of basis for the characteristic-two finite fields. It has the following value:

```
id-characteristic-two-basis OBJECT IDENTIFIER ::= {

characteristic-two-field basisType(3) }
```

The object identifiers **gnBasis**, **tpBasis** and **ppBasis** name the three kinds of basis for characteristic-two finite fields defined in this part of ISO 15782. They have the following values:

```
gnBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 1 }

tpBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 2 }

ppBasis OBJECT IDENTIFIER ::= { id-characteristic-two-basis 3 }
```

NOTE 1 For the finite field F_p, where p is an odd prime, the parameter p is specified by a value of type **Prime-p**.

NOTE 2 For the finite field F_2m, the components of **Characteristic-two** are:

— `m:` degree of the field.

— `basis:` the type of representation used (ONB, TPB, or PPB).

NOTE 3 For a trinomial basis representation of F_2m, `Trinomial` specifies the integer k where $x^m + x^k + 1$ is the reduction polynomial.

NOTE 4 For a pentanomial basis representation of F_2m, the components k1, k2, and k3 of `Pentanomial` specify the integers $k1$, $k2$, and $k3$, respectively, where $xm + x^{k3} + x^{k2} + x^{k1} + 1$ is the reduction polynomial.

E.2.3 Syntax for finite field elements and elliptic curve points

A finite field element shall be represented by a value of type **FieldElement**:

FieldElement ::= OCTET STRING **-- Finite field element**

The value of **FieldElement** shall be the octet string representation of a field element.

An elliptic curve point shall be represented by a value of type **ECPoint**:

ECPoint ::= OCTET STRING **-- Elliptic curve point**

The value of **ECPoint** shall be the octet string representation of an elliptic curve point following the conversion routine in ANS X9.62.

E.2.4 Syntax for elliptic curve parameters

This clause provides syntax for representing elliptic curve parameters.

Elliptic curve parameters shall be represented by a value of type **ECParameters**:

```
Parameters ::= CHOICE {
        ecParameters      ECParameters,
        namedCurve    CURVES.&id({CurveNames}),
        implicitlyCA    NULL
}

CurveNames CURVES ::= {
        { ID c2pnb163v1 }     |    -- J.4.1,   example 1 --
        { ID c2pnb163v2 }     |    -- J.4.1,   example 2 --
        { ID c2pnb163v3 }     |    -- J.4.1,   example 3 --
        { ID c2pnb176w1 }     |    -- J.4.2,   example 1 --
        { ID c2tnb191v1 }     |    -- J.4.3,   example 1 --
        { ID c2tnb191v2 }     |    -- J.4.3,   example 2 --
        { ID c2tnb191v3 }     |    -- J.4.3,   example 3 --
        { ID c2onb191v4 }     |    -- J.4.3,   example 4 --
        { ID c2onb191v5 }     |    -- J.4.3,   example 5 --
        { ID c2pnb208w1 }     |    -- J.4.4,   example 1 --
        { ID c2tnb239v1 }     |    -- J.4.5,   example 1 --
        { ID c2tnb239v2 }     |    -- J.4.5,   example 2 --
        { ID c2tnb239v3 }     |    -- J.4.5,   example 3 --
        { ID c2onb239v4 }     |    -- J.4.5,   example 4 --
        { ID c2onb239v5 }     |    -- J.4.5,   example 5 --
        { ID c2pnb272w1 }     |    -- J.4.6,   example 1 --
        { ID c2pnb304w1 }     |    -- J.4.7,   example 1 --
        { ID c2tnb359v1 }     |    -- J.4.8,   example 1 --
        { ID c2pnb368w1 }     |    -- J.4.9,   example 1 --
        { ID c2tnb431r1 }     |    -- J.4.10, example 1 --
        { ID prime192v1 }     |    -- J.5.1,   example 1 --
        { ID prime192v2 }     |    -- J.5.1,   example 2 --
        { ID prime192v3 }     |    -- J.5.1,   example 3 --
        { ID prime239v1 }     |    -- J.5.2,   example 1 --
        { ID prime239v2 }     |    -- J.5.2,   example 2 --
        { ID prime239v3 }     |    -- J.5.2,   example 3 --
        { ID prime256v1 },           -- J.5.3,   example 1 --
        ...  -- others --
}
```

```
CURVES ::= CLASS {
        &id  OBJECT IDENTIFIER UNIQUE
}
  WITH SYNTAX { ID &id }

ellipticCurve OBJECT IDENTIFIER ::= { ansi-X9-62 curves(3) }

c-TwoCurve OBJECT IDENTIFIER ::= { ellipticCurve characteristicTwo(0) }

primeCurve OBJECT IDENTIFIER ::= { ellipticCurve prime(1) }

c2pnb163v1  OBJECT IDENTIFIER  ::=  { c-TwoCurve  1 }

c2pnb163v2  OBJECT IDENTIFIER  ::=  { c-TwoCurve  2 }

c2pnb163v3  OBJECT IDENTIFIER  ::=  { c-TwoCurve  3 }

c2pnb176w1  OBJECT IDENTIFIER  ::=  { c-TwoCurve  4 }

c2tnb191v1  OBJECT IDENTIFIER  ::=  { c-TwoCurve  5 }

c2tnb191v2  OBJECT IDENTIFIER  ::=  { c-TwoCurve  6 }

c2tnb191v3  OBJECT IDENTIFIER  ::=  { c-TwoCurve  7 }

c2onb191v4  OBJECT IDENTIFIER  ::=  { c-TwoCurve  8 }

c2onb191v5  OBJECT IDENTIFIER  ::=  { c-TwoCurve  9 }

c2pnb208w1  OBJECT IDENTIFIER  ::=  { c-TwoCurve 10 }

c2tnb239v1  OBJECT IDENTIFIER  ::=  { c-TwoCurve 11 }

c2tnb239v2  OBJECT IDENTIFIER  ::=  { c-TwoCurve 12 }

c2tnb239v3  OBJECT IDENTIFIER  ::=  { c-TwoCurve 13 }

c2onb239v4  OBJECT IDENTIFIER  ::=  { c-TwoCurve 14 }

c2onb239v5  OBJECT IDENTIFIER  ::=  { c-TwoCurve 15 }

c2pnb272w1  OBJECT IDENTIFIER  ::=  { c-TwoCurve 16 }

c2pnb304w1  OBJECT IDENTIFIER  ::=  { c-TwoCurve 17 }

c2tnb359v1  OBJECT IDENTIFIER  ::=  { c-TwoCurve 18 }

c2pnb368w1  OBJECT IDENTIFIER  ::=  { c-TwoCurve 19 }

c2tnb431r1  OBJECT IDENTIFIER  ::=  { c-TwoCurve 20 }

prime192v1  OBJECT IDENTIFIER  ::=  { primeCurve  1 }

prime192v2  OBJECT IDENTIFIER  ::=  { primeCurve  2 }

prime192v3  OBJECT IDENTIFIER  ::=  { primeCurve  3 }

prime239v1  OBJECT IDENTIFIER  ::=  { primeCurve  4 }

prime239v2  OBJECT IDENTIFIER  ::=  { primeCurve  5 }

prime239v3  OBJECT IDENTIFIER  ::=  { primeCurve  6 }

prime256v1  OBJECT IDENTIFIER  ::=  { primeCurve  7 }
```

The components of type **ECParameters** have the following meanings:

— **version** specifies the version number of the elliptic curve parameters. It shall have the value 1 for this version of this part of ISO 15782. The notation above creates an **INTEGER** named **ecpVer1** and gives it a value of one. It is used to constrain **version** to a single value.

— **fieldID** identifies the finite field over which the elliptic curve is defined. Finite fields are represented by values of the parameterized type **FieldID**, constrained to the values of the objects defined in the information object set **FieldTypes**.

— **curve** specifies the coefficients a and b of the elliptic curve E. Each coefficient shall be represented as a value of type **FieldElement**, an **OCTET STRING**. **seed** is an optional parameter used to derive the coefficients of a randomly generated elliptic curve.

— **base** specifies the base point P on the elliptic curve. The base point shall be represented as a value of type **ECPoint**, an **OCTET STRING**.

— **ord{ er** specifies the order n of the base point.

— **cofactor** is the integer $h = \#E(F_q)/n$.

E.2.5 Syntax for public keys

This clause provides the syntax for the public keys defined in this part of ISO 15782.

A public key may be represented in a variety of ways using **ASN.1** syntax. When a public key is represented as the **X.509** type **SubjectPublicKeyInfo**, then the public key shall have the following syntax:

```
SubjectPublicKeyInfo ::= SEQUENCE {

        algorithm       AlgorithmIdentifier {{ECPKAlgorithms}},
        subjectPublicKey   BIT STRING
}

AlgorithmIdentifier { ALGORITHM:IOSet } ::= SEQUENCE {
        algorithm ALGORITHM.&id({IOSet}),
        parameters     ALGORITHM.&Type({IOSet}{@algorithm}) OPTIONAL
}

ECPKAlgorithms ALGORITHM ::= {
        ecPublicKeyAlgorithm,
        ...
}

ecPublicKeyAlgorithm ALGORITHM ::= { ECParameters IDENTIFIED BY id-ecPublicKey }

ALGORITHM  ::= TYPE-IDENTIFIER

id-ecPublicKey OBJECT IDENTIFIER ::= { id-publicKeyType 1 }

id-publicKeyType OBJECT IDENTIFIER  ::= { ansi-X9-62 keyType (2) }
```

NOTE 1 The object identifier **id-publicKeyType** represents the tree containing the object identifiers for each public key. It has the following value:

```
id-public-key-type OBJECT IDENTIFIER  ::= {ansi-X9-62 2}
```

NOTE 2 The elliptic curve public key (an **ECPoint** which is an **OCTET STRING**) is mapped to a **subjectPublicKey** (a **BIT STRING**) as follows: the most significant bit of the **OCTET STRING** becomes the most significant bit of the **BIT STRING**, etc.; the least significant bit of the **OCTET STRING** becomes the least significant bit of the **BIT STRING**.

E.2.6 Syntax for digital signatures

This clause provides the syntax for the digital signatures defined in this part of ISO 15782.

A signature may be represented in a variety of ways using the ASN.1 notation. The X.509 certificate and CRL types include an ASN.1 algorithm object identifier to identify the signature type and format. When ECDSA and SHA-1 are used to sign an X.509 certificate or CRL, the signature shall be identified by the value ecdsa-with-SHA1, as defined below:

id-ecSigType OBJECT IDENTIFIER ::= { ansi-X9-62 signatures(4) }

ecdsa-with-SHA1 OBJECT IDENTIFIER ::= { id-ecSigType 1 }

When the **ecdsa-with-SHA1** OID appears in the algorithm field of the ASN.1 type **AlgorithmIdentifier**, and the parameters field is a value of type **NULL**, the ECDSA parameters for signature verification must be obtained from other sources, such as the **subjectPublicKeyInfo** field of the certificate of the issuer.

When a digital signature is identified by the OID **ecdsa-with-SHA1**, the digital signature shall be ASN.1 encoded using the following syntax:

```
ECDSA-Sig-Value ::= SEQUENCE {
        r       INTEGER,
        s       INTEGER
}
```

X.509 certificates and CRLs represent signatures as a bit string. Where a certificate or CRL is signed with ECDSA and SHA-1, the entire encoding of a value of ASN.1 type **ECDSA-Sig-Value** shall be the value of the bit string."

E.3 RSA public keys

An RSA public key is conveyed as the DER encoding of the modulus and public exponent:

```
RSAPublicKey ::= SEQUENCE {
        n       INTEGER,
        e       INTEGER
}
```

An RSA signature (an **INTEGER**) is conveyed in a **BIT STRING** as follows: the most significant bit of the **INTEGER** becomes the most significant bit of the **BIT STRING**, and the least significant bit of the **INTEGER** becomes the least significant bit of the **BIT STRING**.

There are no parameters for RSA:

```
rsa-signature ALGORITHM ::= { NULL IDENTIFIED BY id-rsa-signature }

rsa-signature-with-sha1 ALGORITHM ::=
        { NULL IDENTIFIED BY id-rsa-signature-with sha1 }
```