Network Working Group
Request for Comments: 2459
Category: Standards Track

R. Housley
SPYRUS
W. Ford
VeriSign
W. Polk
NIST
D. Solo
Citicorp

January 1999

Internet X.509 Public Key Infrastructure Certificate and CRL Profile

Status of this Memo

Copyright Notice

Abstract

This memo profiles the X.509 v3 certificate and X.509 v2 CRL for use in the Internet. An overview of the approach and model are provided as an introduction. The X.509 v3 certificate format is described in detail, with additional information regarding the format and semantics of Internet name forms (e.g., IP addresses). Standard certificate extensions are described and one new Internet-specific extension is defined. A required set of certificate extensions is specified. The X.509 v2 CRL format is described and a required extension set is defined as well. An algorithm for X.509 certificate path validation is described. Supplemental information is provided describing the format of public keys and digital signatures in X.509 certificates for common Internet public key encryption algorithms (i.e., RSA, DSA, and Diffie-Hellman). ASN.1 modules and examples are provided in the appendices.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Please send comments on this document to the ietf-pkix@imc.org mail list.

Table of Contents

1 Introduction

This specification is one part of a family of standards for the X.509 Public Key Infrastructure (PKI) for the Internet. This specification is a standalone document; implementations of this standard may proceed independent from the other parts.

This specification profiles the format and semantics of certificates and certificate revocation lists for the Internet PKI. Procedures are described for processing of certification paths in the Internet environment. Encoding rules are provided for popular cryptographic algorithms. Finally, ASN.1 modules are provided in the appendices for all data structures defined or referenced.

The specification describes the requirements which inspire the creation of this document and the assumptions which affect its scope in Section 2. Section 3 presents an architectural model and describes its relationship to previous IETF and ISO/IEC/ITU standards. In particular, this document's relationship with the IETF PEM specifications and the ISO/IEC/ITU X.509 documents are described.

The specification profiles the X.509 version 3 certificate in Section 4, and the X.509 version 2 certificate revocation list (CRL) in Section 5. The profiles include the identification of ISO/IEC/ITU and ANSI extensions which may be useful in the Internet PKI. The profiles are presented in the 1988 Abstract Syntax Notation One (ASN.1) rather than the 1994 syntax used in the ISO/IEC/ITU standards.

This specification also includes path validation procedures in Section 6. These procedures are based upon the ISO/IEC/ITU definition, but the presentation assumes one or more self-signed trusted CA certificates. Implementations are required to derive the same results but are not required to use the specified procedures.

Section 7 of the specification describes procedures for identification and encoding of public key materials and digital signatures. Implementations are not required to use any particular cryptographic algorithms. However, conforming implementations which use the identified algorithms are required to identify and encode the public key materials and digital signatures as described.

Finally, four appendices are provided to aid implementers. Appendix A contains all ASN.1 structures defined or referenced within this specification. As above, the material is presented in the 1988 Abstract Syntax Notation One (ASN.1) rather than the 1994 syntax. Appendix B contains the same information in the 1994 ASN.1 notation as a service to implementers using updated toolsets. However, Appendix A takes precedence in case of conflict. Appendix C contains

notes on less familiar features of the ASN.1 notation used within this specification. Appendix D contains examples of a conforming certificate and a conforming CRL.

2 Requirements and Assumptions

The goal of this specification is to develop a profile to facilitate the use of X.509 certificates within Internet applications for those communities wishing to make use of X.509 technology. Such applications may include WWW, electronic mail, user authentication, and IPsec. In order to relieve some of the obstacles to using X.509 certificates, this document defines a profile to promote the development of certificate management systems; development of application tools; and interoperability determined by policy.

Some communities will need to supplement, or possibly replace, this profile in order to meet the requirements of specialized application domains or environments with additional authorization, assurance, or operational requirements. However, for basic applications, common representations of frequently used attributes are defined so that application developers can obtain necessary information without regard to the issuer of a particular certificate or certificate revocation list (CRL).

A certificate user should review the certificate policy generated by the certification authority (CA) before relying on the authentication or non-repudiation services associated with the public key in a particular certificate. To this end, this standard does not prescribe legally binding rules or duties.

As supplemental authorization and attribute management tools emerge, such as attribute certificates, it may be appropriate to limit the authenticated attributes that are included in a certificate. These other management tools may provide more appropriate methods of conveying many authenticated attributes.

2.1 Communication and Topology

The users of certificates will operate in a wide range of environments with respect to their communication topology, especially users of secure electronic mail. This profile supports users without high bandwidth, real-time IP connectivity, or high connection availability. In addition, the profile allows for the presence of firewall or other filtered communication.

This profile does not assume the deployment of an X.500 Directory system. The profile does not prohibit the use of an X.500 Directory, but other means of distributing certificates and certificate revocation lists (CRLs) may be used.

2.2 Acceptability Criteria

The goal of the Internet Public Key Infrastructure (PKI) is to meet the needs of deterministic, automated identification, authentication, access control, and authorization functions. Support for these services determines the attributes contained in the certificate as well as the ancillary control information in the certificate such as policy data and certification path constraints.

2.3 User Expectations

Users of the Internet PKI are people and processes who use client software and are the subjects named in certificates. These uses include readers and writers of electronic mail, the clients for WWW browsers, WWW servers, and the key manager for IPsec within a router. This profile recognizes the limitations of the platforms these users employ and the limitations in sophistication and attentiveness of the users themselves. This manifests itself in minimal user configuration responsibility (e.g., trusted CA keys, rules), explicit platform usage constraints within the certificate, certification path constraints which shield the user from many malicious actions, and applications which sensibly automate validation functions.

2.4 Administrator Expectations

As with user expectations, the Internet PKI profile is structured to support the individuals who generally operate CAs. Providing administrators with unbounded choices increases the chances that a subtle CA administrator mistake will result in broad compromise. Also, unbounded choices greatly complicate the software that shall process and validate the certificates created by the CA.

3 Overview of Approach

Following is a simplified view of the architectural model assumed by the PKIX specifications.

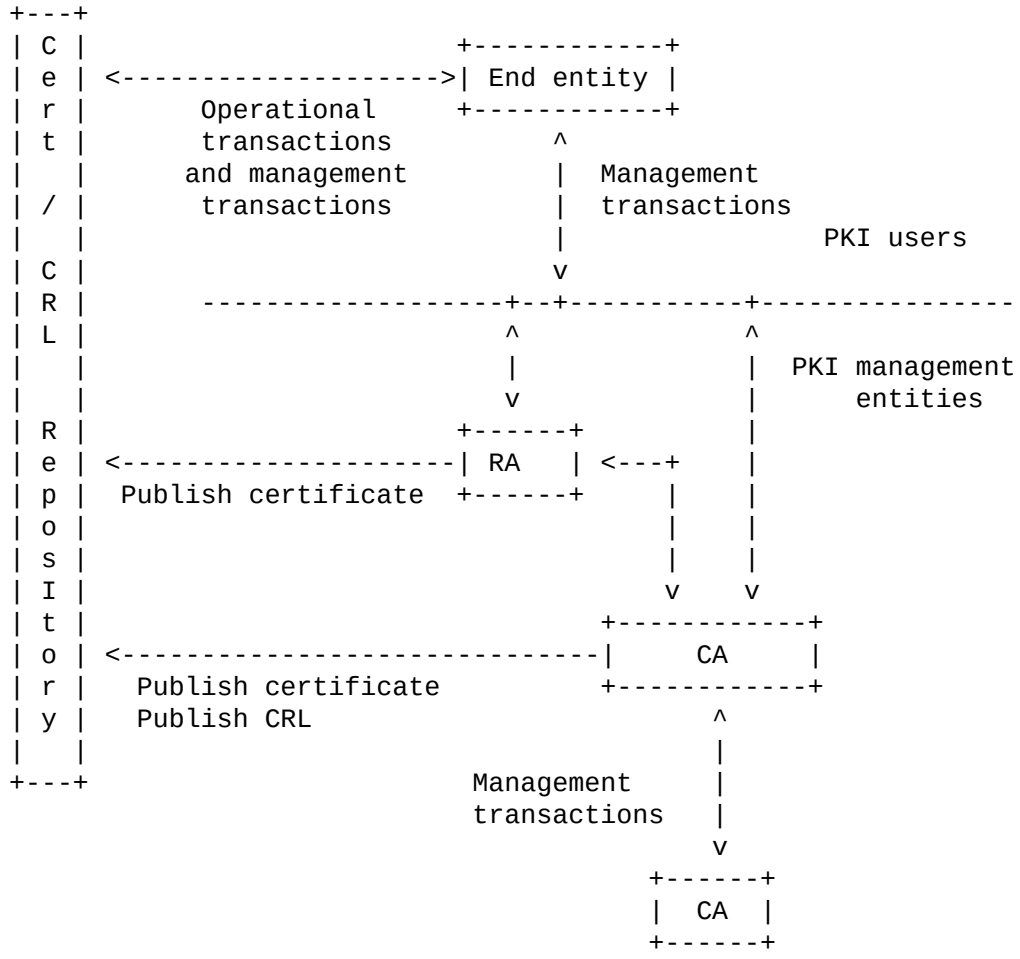

Figure 1 - PKI Entities

The components in this model are:

end entity: user of PKI certificates and/or end user system that is the subject of a certificate;
CA: certification authority;
RA: registration authority, i.e., an optional system to which a CA delegates certain management functions;
repository: a system or collection of distributed systems that store certificates and CRLs and serves as a means of distributing these certificates and CRLs to end entities.

3.1 X.509 Version 3 Certificate

Users of a public key shall be confident that the associated private key is owned by the correct remote subject (person or system) with which an encryption or digital signature mechanism will be used. This confidence is obtained through the use of public key certificates, which are data structures that bind public key values to subjects. The binding is asserted by having a trusted CA digitally sign each certificate. The CA may base this assertion upon technical means (a.k.a., proof of posession through a challenge-response protocol), presentation of the private key, or on an assertion by the subject. A certificate has a limited valid lifetime which is indicated in its signed contents. Because a certificate's signature and timeliness can be independently checked by a certificate-using client, certificates can be distributed via untrusted communications and server systems, and can be cached in unsecured storage in certificate-using systems.

ITU-T X.509 (formerly CCITT X.509) or ISO/IEC/ITU 9594-8, which was first published in 1988 as part of the X.500 Directory recommendations, defines a standard certificate format [X.509]. The certificate format in the 1988 standard is called the version 1 (v1) format. When X.500 was revised in 1993, two more fields were added, resulting in the version 2 (v2) format. These two fields may be used to support directory access control.

The Internet Privacy Enhanced Mail (PEM) RFCs, published in 1993, include specifications for a public key infrastructure based on X.509 v1 certificates [RFC 1422]. The experience gained in attempts to deploy RFC 1422 made it clear that the v1 and v2 certificate formats are deficient in several respects. Most importantly, more fields were needed to carry information which PEM design and implementation experience has proven necessary. In response to these new requirements, ISO/IEC/ITU and ANSI X9 developed the X.509 version 3 (v3) certificate format. The v3 format extends the v2 format by adding provision for additional extension fields. Particular extension field types may be specified in standards or may be defined and registered by any organization or community. In June 1996, standardization of the basic v3 format was completed [X.509].

ISO/IEC/ITU and ANSI X9 have also developed standard extensions for use in the v3 extensions field [X.509][X9.55]. These extensions can convey such data as additional subject identification information, key attribute information, policy information, and certification path constraints.

However, the ISO/IEC/ITU and ANSI X9 standard extensions are very broad in their applicability. In order to develop interoperable implementations of X.509 v3 systems for Internet use, it is necessary to specify a profile for use of the X.509 v3 extensions tailored for the Internet. It is one goal of this document to specify a profile for Internet WWW, electronic mail, and IPsec applications. Environments with additional requirements may build on this profile or may replace it.

3.2 Certification Paths and Trust

A user of a security service requiring knowledge of a public key generally needs to obtain and validate a certificate containing the required public key. If the public-key user does not already hold an assured copy of the public key of the CA that signed the certificate, the CA's name, and related information (such as the validity period or name constraints), then it might need an additional certificate to obtain that public key. In general, a chain of multiple certificates may be needed, comprising a certificate of the public key owner (the end entity) signed by one CA, and zero or more additional certificates of CAs signed by other CAs. Such chains, called certification paths, are required because a public key user is only initialized with a limited number of assured CA public keys.

There are different ways in which CAs might be configured in order for public key users to be able to find certification paths. For PEM, RFC 1422 defined a rigid hierarchical structure of CAs. There are three types of PEM certification authority:

(a) Internet Policy Registration Authority (IPRA): This authority, operated under the auspices of the Internet Society, acts as the root of the PEM certification hierarchy at level 1. It issues certificates only for the next level of authorities, PCAs. All certification paths start with the IPRA.

(b) Policy Certification Authorities (PCAs): PCAs are at level 2 of the hierarchy, each PCA being certified by the IPRA. A PCA shall establish and publish a statement of its policy with respect to certifying users or subordinate certification authorities. Distinct PCAs aim to satisfy different user needs. For example, one PCA (an organizational PCA) might support the general electronic mail needs of commercial organizations, and another PCA (a high-assurance PCA) might have a more stringent policy designed for satisfying legally binding digital signature requirements.

(c) Certification Authorities (CAs): CAs are at level 3 of the hierarchy and can also be at lower levels. Those at level 3 are certified by PCAs. CAs represent, for example, particular organizations, particular organizational units (e.g., departments, groups, sections), or particular geographical areas.

RFC 1422 furthermore has a name subordination rule which requires that a CA can only issue certificates for entities whose names are subordinate (in the X.500 naming tree) to the name of the CA itself. The trust associated with a PEM certification path is implied by the PCA name. The name subordination rule ensures that CAs below the PCA are sensibly constrained as to the set of subordinate entities they can certify (e.g., a CA for an organization can only certify entities in that organization's name tree). Certificate user systems are able to mechanically check that the name subordination rule has been followed.

The RFC 1422 uses the X.509 v1 certificate formats. The limitations of X.509 v1 required imposition of several structural restrictions to clearly associate policy information or restrict the utility of certificates. These restrictions included:

(a) a pure top-down hierarchy, with all certification paths starting from IPRA;

(b) a naming subordination rule restricting the names of a CA's subjects; and

(c) use of the PCA concept, which requires knowledge of individual PCAs to be built into certificate chain verification logic. Knowledge of individual PCAs was required to determine if a chain could be accepted.

With X.509 v3, most of the requirements addressed by RFC 1422 can be addressed using certificate extensions, without a need to restrict the CA structures used. In particular, the certificate extensions relating to certificate policies obviate the need for PCAs and the constraint extensions obviate the need for the name subordination rule. As a result, this document supports a more flexible architecture, including:

(a) Certification paths may start with a public key of a CA in a user's own domain, or with the public key of the top of a hierarchy. Starting with the public key of a CA in a user's own domain has certain advantages. In some environments, the local domain is the most trusted.

(b) Name constraints may be imposed through explicit inclusion of a name constraints extension in a certificate, but are not required.

(c) Policy extensions and policy mappings replace the PCA concept, which permits a greater degree of automation. The application can determine if the certification path is acceptable based on the contents of the certificates instead of a priori knowledge of PCAs. This permits automation of certificate chain processing.

3.3 Revocation

When a certificate is issued, it is expected to be in use for its entire validity period. However, various circumstances may cause a certificate to become invalid prior to the expiration of the validity period. Such circumstances include change of name, change of association between subject and CA (e.g., an employee terminates employment with an organization), and compromise or suspected compromise of the corresponding private key. Under such circumstances, the CA needs to revoke the certificate.

X.509 defines one method of certificate revocation. This method involves each CA periodically issuing a signed data structure called a certificate revocation list (CRL). A CRL is a time stamped list identifying revoked certificates which is signed by a CA and made freely available in a public repository. Each revoked certificate is identified in a CRL by its certificate serial number. When a certificate-using system uses a certificate (e.g., for verifying a remote user's digital signature), that system not only checks the certificate signature and validity but also acquires a suitably-recent CRL and checks that the certificate serial number is not on that CRL. The meaning of "suitably-recent" may vary with local policy, but it usually means the most recently-issued CRL. A CA issues a new CRL on a regular periodic basis (e.g., hourly, daily, or weekly). An entry is added to the CRL as part of the next update following notification of revocation. An entry may be removed from the CRL after appearing on one regularly scheduled CRL issued beyond the revoked certificate's validity period.

An advantage of this revocation method is that CRLs may be distributed by exactly the same means as certificates themselves, namely, via untrusted communications and server systems.

One limitation of the CRL revocation method, using untrusted communications and servers, is that the time granularity of revocation is limited to the CRL issue period. For example, if a revocation is reported now, that revocation will not be reliably

notified to certificate-using systems until the next periodic CRL is issued -- this may be up to one hour, one day, or one week depending on the frequency that the CA issues CRLs.

As with the X.509 v3 certificate format, in order to facilitate interoperable implementations from multiple vendors, the X.509 v2 CRL format needs to be profiled for Internet use. It is one goal of this document to specify that profile. However, this profile does not require CAs to issue CRLs. Message formats and protocols supporting on-line revocation notification may be defined in other PKIX specifications. On-line methods of revocation notification may be applicable in some environments as an alternative to the X.509 CRL. On-line revocation checking may significantly reduce the latency between a revocation report and the distribution of the information to relying parties. Once the CA accepts the report as authentic and valid, any query to the on-line service will correctly reflect the certificate validation impacts of the revocation. However, these methods impose new security requirements; the certificate validator shall trust the on-line validation service while the repository does not need to be trusted.

3.4 Operational Protocols

Operational protocols are required to deliver certificates and CRLs (or status information) to certificate using client systems. Provision is needed for a variety of different means of certificate and CRL delivery, including distribution procedures based on LDAP, HTTP, FTP, and X.500. Operational protocols supporting these functions are defined in other PKIX specifications. These specifications may include definitions of message formats and procedures for supporting all of the above operational environments, including definitions of or references to appropriate MIME content types.

3.5 Management Protocols

Management protocols are required to support on-line interactions between PKI user and management entities. For example, a management protocol might be used between a CA and a client system with which a key pair is associated, or between two CAs which cross-certify each other. The set of functions which potentially need to be supported by management protocols include:

(a) registration: This is the process whereby a user first makes itself known to a CA (directly, or through an RA), prior to that CA issuing a certificate or certificates for that user.

(b) initialization: Before a client system can operate securely it is necessary to install key materials which have the appropriate relationship with keys stored elsewhere in the infrastructure. For example, the client needs to be securely initialized with the public key and other assured information of the trusted CA(s), to be used in validating certificate paths. Furthermore, a client typically needs to be initialized with its own key pair(s).

(c) certification: This is the process in which a CA issues a certificate for a user's public key, and returns that certificate to the user's client system and/or posts that certificate in a repository.

(d) key pair recovery: As an option, user client key materials (e.g., a user's private key used for encryption purposes) may be backed up by a CA or a key backup system. If a user needs to recover these backed up key materials (e.g., as a result of a forgotten password or a lost key chain file), an on-line protocol exchange may be needed to support such recovery.

(e) key pair update: All key pairs need to be updated regularly, i.e., replaced with a new key pair, and new certificates issued.

(f) revocation request: An authorized person advises a CA of an abnormal situation requiring certificate revocation.

(g) cross-certification: Two CAs exchange information used in establishing a cross-certificate. A cross-certificate is a certificate issued by one CA to another CA which contains a CA signature key used for issuing certificates.

Note that on-line protocols are not the only way of implementing the above functions. For all functions there are off-line methods of achieving the same result, and this specification does not mandate use of on-line protocols. For example, when hardware tokens are used, many of the functions may be achieved as part of the physical token delivery. Furthermore, some of the above functions may be combined into one protocol exchange. In particular, two or more of the registration, initialization, and certification functions can be combined into one protocol exchange.

The PKIX series of specifications may define a set of standard message formats supporting the above functions in future specifications. In that case, the protocols for conveying these messages in different environments (e.g., on-line, file transfer, e-mail, and WWW) will also be described in those specifications.

4 Certificate and Certificate Extensions Profile

This section presents a profile for public key certificates that will foster interoperability and a reusable PKI. This section is based upon the X.509 v3 certificate format and the standard certificate extensions defined in [X.509]. The ISO/IEC/ITU documents use the 1993 version of ASN.1; while this document uses the 1988 ASN.1 syntax, the encoded certificate and standard extensions are equivalent. This section also defines private extensions required to support a PKI for the Internet community.

Certificates may be used in a wide range of applications and environments covering a broad spectrum of interoperability goals and a broader spectrum of operational and assurance requirements. The goal of this document is to establish a common baseline for generic applications requiring broad interoperability and limited special purpose requirements. In particular, the emphasis will be on supporting the use of X.509 v3 certificates for informal Internet electronic mail, IPsec, and WWW applications.

4.1 Basic Certificate Fields

The X.509 v3 certificate basic syntax is as follows. For signature calculation, the certificate is encoded using the ASN.1 distinguished encoding rules (DER) [X.208]. ASN.1 DER encoding is a tag, length, value encoding system for each element.

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

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

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

CertificateSerialNumber  ::=  INTEGER

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

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

UniqueIdentifier  ::=  BIT STRING

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

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

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

The following items describe the X.509 v3 certificate for use in the Internet.

4.1.1 Certificate Fields

The Certificate is a SEQUENCE of three required fields. The fields are described in detail in the following subsections.

4.1.1.1 tbsCertificate

The field contains the names of the subject and issuer, a public key associated with the subject, a validity period, and other associated information. The fields are described in detail in section 4.1.2; the tbscertificate may also include extensions which are described in section 4.2.

4.1.1.2 signatureAlgorithm

The signatureAlgorithm field contains the identifier for the cryptographic algorithm used by the CA to sign this certificate. Section 7.2 lists the supported signature algorithms.

An algorithm identifier is defined by the following ASN.1 structure:

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

The algorithm identifier is used to identify a cryptographic algorithm. The OBJECT IDENTIFIER component identifies the algorithm (such as DSA with SHA-1). The contents of the optional parameters field will vary according to the algorithm identified. Section 7.2 lists the supported algorithms for this specification.

This field MUST contain the same algorithm identifier as the signature field in the sequence tbsCertificate (see sec. 4.1.2.3).

4.1.1.3 signatureValue

The signatureValue field contains a digital signature computed upon the ASN.1 DER encoded tbsCertificate. The ASN.1 DER encoded tbsCertificate is used as the input to the signature function. This signature value is then ASN.1 encoded as a BIT STRING and included in the Certificate's signature field. The details of this process are specified for each of the supported algorithms in Section 7.2.

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

4.1.2 TBSCertificate

The sequence TBSCertificate contains information associated with the subject of the certificate and the CA who issued it. Every TBSCertificate contains the names of the subject and issuer, a public key associated with the subject, a validity period, a version number, and a serial number; some may contain optional unique identifier fields. The remainder of this section describes the syntax and semantics of these fields. A TBSCertificate may also include extensions. Extensions for the Internet PKI are described in Section 4.2.

4.1.2.1 Version

This field describes the version of the encoded certificate. When extensions are used, as expected in this profile, use X.509 version 3 (value is 2). If no extensions are present, but a UniqueIdentifier is present, use version 2 (value is 1). If only basic fields are present, use version 1 (the value is omitted from the certificate as the default value).

Implementations SHOULD be prepared to accept any version certificate. At a minimum, conforming implementations MUST recognize version 3 certificates.

Generation of version 2 certificates is not expected by implementations based on this profile.

4.1.2.2 Serial number

The serial number is an integer assigned by the CA to each certificate. It MUST be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate).

4.1.2.3 Signature

This field contains the algorithm identifier for the algorithm used by the CA to sign the certificate.

This field MUST contain the same algorithm identifier as the signatureAlgorithm field in the sequence Certificate (see sec. 4.1.1.2). The contents of the optional parameters field will vary according to the algorithm identified. Section 7.2 lists the supported signature algorithms.

4.1.2.4 Issuer

The issuer field identifies the entity who has signed and issued the certificate. The issuer field MUST contain a non-empty distinguished name (DN). The issuer field is defined as the X.501 type Name. [X.501] Name is defined by the following ASN.1 structures:

```
Name ::= CHOICE {
  RDNSequence }

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::=
  SET OF AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
  type     AttributeType,
  value    AttributeValue }

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY DEFINED BY AttributeType
```

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

The Name describes a hierarchical name composed of attributes, such as country name, and corresponding values, such as US. The type of the component AttributeValue is determined by the AttributeType; in general it will be a DirectoryString.

The DirectoryString type is defined as a choice of PrintableString, TeletexString, BMPString, UTF8String, and UniversalString. The UTF8String encoding is the preferred encoding, and all certificates issued after December 31, 2003 MUST use the UTF8String encoding of DirectoryString (except as noted below). Until that date, conforming CAs MUST choose from the following options when creating a distinguished name, including their own:

(a) if the character set is sufficient, the string MAY be represented as a PrintableString;

(b) failing (a), if the BMPString character set is sufficient the string MAY be represented as a BMPString; and

(c) failing (a) and (b), the string MUST be represented as a UTF8String. If (a) or (b) is satisfied, the CA MAY still choose to represent the string as a UTF8String.

Exceptions to the December 31, 2003 UTF8 encoding requirements are as follows:

(a) CAs MAY issue "name rollover" certificates to support an orderly migration to UTF8String encoding. Such certificates would include the CA's UTF8String encoded name as issuer and and the old name encoding as subject, or vice-versa.

(b) As stated in section 4.1.2.6, the subject field MUST be populated with a non-empty distinguished name matching the contents of the issuer field in all certificates issued by the subject CA regardless of encoding.

The TeletexString and UniversalString are included for backward compatibility, and should not be used for certificates for new subjects. However, these types may be used in certificates where the name was previously established. Certificate users SHOULD be prepared to receive certificates with these types.

In addition, many legacy implementations support names encoded in the ISO 8859-1 character set (Latin1String) but tag them as TeletexString. The Latin1String includes characters used in Western European countries which are not part of the TeletexString charcter set. Implementations that process TeletexString SHOULD be prepared to handle the entire ISO 8859-1 character set.[ISO 8859-1]

As noted above, distinguished names are composed of attributes. This specification does not restrict the set of attribute types that may appear in names. However, conforming implementations MUST be prepared to receive certificates with issuer names containing the set of attribute types defined below. This specification also recommends support for additional attribute types.

Standard sets of attributes have been defined in the X.500 series of specifications.[X.520] Implementations of this specification MUST be prepared to receive the following standard attribute types in issuer names: country, organization, organizational-unit, distinguished name qualifier, state or province name, and common name (e.g., "Susan Housley"). In addition, implementations of this specification SHOULD be prepared to receive the following standard attribute types in issuer names: locality, title, surname, given name, initials, and generation qualifier (e.g., "Jr.", "3rd", or "IV"). The syntax and associated object identifiers (OIDs) for these attribute types are provided in the ASN.1 modules in Appendices A and B.

In addition, implementations of this specification MUST be prepared to receive the domainComponent attribute, as defined in [RFC 2247]. The Domain (Nameserver) System (DNS) provides a hierarchical resource labeling system. This attribute provides is a convenient mechanism for organizations that wish to use DNs that parallel their DNS names. This is not a replacement for the dNSName component of the alternative name field. Implementations are not required to convert such names into DNS names. The syntax and associated OID for this attribute type is provided in the ASN.1 modules in Appendices A and B.

Certificate users MUST be prepared to process the issuer distinguished name and subject distinguished name (see sec. 4.1.2.6) fields to perform name chaining for certification path validation (see section 6). Name chaining is performed by matching the issuer distinguished name in one certificate with the subject name in a CA certificate.

This specification requires only a subset of the name comparison functionality specified in the X.500 series of specifications. The requirements for conforming implementations are as follows:

(a) attribute values encoded in different types (e.g., PrintableString and BMPString) may be assumed to represent different strings;

(b) attribute values in types other than PrintableString are case sensitive (this permits matching of attribute values as binary objects);

(c) attribute values in PrintableString are not case sensitive (e.g., "Marianne Swanson" is the same as "MARIANNE SWANSON"); and

(d) attribute values in PrintableString are compared after removing leading and trailing white space and converting internal substrings of one or more consecutive white space characters to a single space.

These name comparison rules permit a certificate user to validate certificates issued using languages or encodings unfamiliar to the certificate user.

In addition, implementations of this specification MAY use these comparison rules to process unfamiliar attribute types for name chaining. This allows implementations to process certificates with unfamiliar attributes in the issuer name.

Note that the comparison rules defined in the X.500 series of specifications indicate that the character sets used to encode data in distinguished names are irrelevant. The characters themselves are compared without regard to encoding. Implementations of the profile are permitted to use the comparison algorithm defined in the X.500 series. Such an implementation will recognize a superset of name matches recognized by the algorithm specified above.

4.1.2.5 Validity

The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate. The field is represented as a SEQUENCE of two dates: the date on which the certificate validity period begins (notBefore) and the date on which the certificate validity period ends (notAfter). Both notBefore and notAfter may be encoded as UTCTime or GeneralizedTime.

CAs conforming to this profile MUST always encode certificate validity dates through the year 2049 as UTCTime; certificate validity dates in 2050 or later MUST be encoded as GeneralizedTime.

4.1.2.5.1 UTCTime

The universal time type, UTCTime, is a standard ASN.1 type intended for international applications where local time alone is not adequate. UTCTime specifies the year through the two low order digits and time is specified to the precision of one minute or one second. UTCTime includes either Z (for Zulu, or Greenwich Mean Time) or a time differential.

For the purposes of this profile, UTCTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYMMDDHHMMSSZ), even where the number of seconds is zero. Conforming systems MUST interpret the year field (YY) as follows:

> Where YY is greater than or equal to 50, the year shall be interpreted as 19YY; and
>
> Where YY is less than 50, the year shall be interpreted as 20YY.

4.1.2.5.2 GeneralizedTime

The generalized time type, GeneralizedTime, is a standard ASN.1 type for variable precision representation of time. Optionally, the GeneralizedTime field can include a representation of the time differential between local and Greenwich Mean Time.

For the purposes of this profile, GeneralizedTime values MUST be expressed Greenwich Mean Time (Zulu) and MUST include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero. GeneralizedTime values MUST NOT include fractional seconds.

4.1.2.6 Subject

The subject field identifies the entity associated with the public key stored in the subject public key field. The subject name may be carried in the subject field and/or the subjectAltName extension. If the subject is a CA (e.g., the basic constraints extension, as discussed in 4.2.1.10, is present and the value of cA is TRUE,) then the subject field MUST be populated with a non-empty distinguished name matching the contents of the issuer field (see sec. 4.1.2.4) in all certificates issued by the subject CA. If subject naming information is present only in the subjectAltName extension (e.g., a key bound only to an email address or URI), then the subject name MUST be an empty sequence and the subjectAltName extension MUST be critical.

Where it is non-empty, the subject field MUST contain an X.500 distinguished name (DN). The DN MUST be unique for each subject entity certified by the one CA as defined by the issuer name field. A CA may issue more than one certificate with the same DN to the same subject entity.

The subject name field is defined as the X.501 type Name. Implementation requirements for this field are those defined for the issuer field (see sec. 4.1.2.4). When encoding attribute values of type DirectoryString, the encoding rules for the issuer field MUST be implemented. Implementations of this specification MUST be prepared to receive subject names containing the attribute types required for the issuer field. Implementations of this specification SHOULD be prepared to receive subject names containing the recommended attribute types for the issuer field. The syntax and associated object identifiers (OIDs) for these attribute types are provided in the ASN.1 modules in Appendices A and B. Implementations of this specification MAY use these comparison rules to process unfamiliar attribute types (i.e., for name chaining). This allows implementations to process certificates with unfamiliar attributes in the subject name.

In addition, legacy implementations exist where an RFC 822 name is embedded in the subject distinguished name as an EmailAddress attribute. The attribute value for EmailAddress is of type IA5String to permit inclusion of the character '@', which is not part of the PrintableString character set. EmailAddress attribute values are not case sensitive (e.g., "fanfeedback@redsox.com" is the same as "FANFEEDBACK@REDSOX.COM").

Conforming implementations generating new certificates with electronic mail addresses MUST use the rfc822Name in the subject alternative name field (see sec. 4.2.1.7) to describe such identities. Simultaneous inclusion of the EmailAddress attribute in the subject distinguished name to support legacy implementations is deprecated but permitted.

4.1.2.7 Subject Public Key Info

This field is used to carry the public key and identify the algorithm with which the key is used. The algorithm is identified using the AlgorithmIdentifier structure specified in section 4.1.1.2. The object identifiers for the supported algorithms and the methods for encoding the public key materials (public key and parameters) are specified in section 7.3.

4.1.2.8 Unique Identifiers

These fields may only appear if the version is 2 or 3 (see sec. 4.1.2.1). The subject and issuer unique identifiers are present in the certificate to handle the possibility of reuse of subject and/or issuer names over time. This profile recommends that names not be reused for different entities and that Internet certificates not make use of unique identifiers. CAs conforming to this profile SHOULD NOT generate certificates with unique identifiers. Applications conforming to this profile SHOULD be capable of parsing unique identifiers and making comparisons.

4.1.2.9 Extensions

This field may only appear if the version is 3 (see sec. 4.1.2.1). If present, this field is a SEQUENCE of one or more certificate extensions. The format and content of certificate extensions in the Internet PKI is defined in section 4.2.

4.2 Standard Certificate Extensions

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

Each extension includes an OID and an ASN.1 structure. When an extension appears in a certificate, the OID appears as the field extnID and the corresponding ASN.1 encoded structure is the value of the octet string extnValue. Only one instance of a particular extension may appear in a particular certificate. For example, a certificate may contain only one authority key identifier extension (see sec. 4.2.1.1). An extension includes the boolean critical, with a default value of FALSE. The text for each extension specifies the acceptable values for the critical field.

Conforming CAs MUST support key identifiers (see sec. 4.2.1.1 and 4.2.1.2), basic constraints (see sec. 4.2.1.10), key usage (see sec. 4.2.1.3), and certificate policies (see sec. 4.2.1.5) extensions. If the CA issues certificates with an empty sequence for the subject field, the CA MUST support the subject alternative name extension (see sec. 4.2.1.7). Support for the remaining extensions is OPTIONAL. Conforming CAs may support extensions that are not identified within this specification; certificate issuers are cautioned that marking such extensions as critical may inhibit interoperability.

At a minimum, applications conforming to this profile MUST recognize the extensions which must or may be critical in this specification. These extensions are: key usage (see sec. 4.2.1.3), certificate policies (see sec. 4.2.1.5), the subject alternative name (see sec. 4.2.1.7), basic constraints (see sec. 4.2.1.10), name constraints (see sec. 4.2.1.11), policy constraints (see sec. 4.2.1.12), and extended key usage (see sec. 4.2.1.13).

In addition, this profile RECOMMENDS application support for the authority and subject key identifier (see sec. 4.2.1.1 and 4.2.1.2) extensions.

4.2.1 Standard Extensions

This section identifies standard certificate extensions defined in [X.509] for use in the Internet PKI. Each extension is associated with an OID defined in [X.509]. These OIDs are members of the id-ce arc, which is defined by the following:

```
id-ce   OBJECT IDENTIFIER ::=  {joint-iso-ccitt(2) ds(5) 29}
```

4.2.1.1 Authority Key Identifier

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

The keyIdentifier field of the authorityKeyIdentifier extension MUST be included in all certificates generated by conforming CAs to facilitate chain building. There is one exception; where a CA distributes its public key in the form of a "self-signed" certificate, the authority key identifier may be omitted. In this case, the subject and authority key identifiers would be identical.

The value of the keyIdentifier field SHOULD be derived from the public key used to verify the certificate's signature or a method that generates unique values. Two common methods for generating key identifiers from the public key are described in (sec. 4.2.1.2). One common method for generating unique values isdescribed in (sec. 4.2.1.2). Where a key identifier has not been previously established, this specification recommends use of one of these methods for generating keyIdentifiers.

This profile recommends support for the key identifier method by all certificate users.

This extension MUST NOT be marked critical.

```
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 35 }

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

KeyIdentifier ::= OCTET STRING
```

4.2.1.2 Subject Key Identifier

The subject key identifier extension provides a means of identifying certificates that contain a particular public key.

To facilitate chain building, this extension MUST appear in all conforming CA certificates, that is, all certificates including the basic constraints extension (see sec. 4.2.1.10) where the value of cA is TRUE. The value of the subject key identifier MUST be the value placed in the key identifier field of the Authority Key Identifier extension (see sec. 4.2.1.1) of certificates issued by the subject of this certificate.

For CA certificates, subject key identifiers SHOULD be derived from the public key or a method that generates unique values. Two common methods for generating key identifiers from the public key are:

(1) The keyIdentifier is composed of the 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey (excluding the tag, length, and number of unused bits).

(2) The keyIdentifier is composed of a four bit type field with the value 0100 followed by the least significant 60 bits of the SHA-1 hash of the value of the BIT STRING subjectPublicKey.

One common method for generating unique values is a monotomically increasing sequence of integers.

For end entity certificates, the subject key identifier extension provides a means for identifying certificates containing the particular public key used in an application. Where an end entity has obtained multiple certificates, especially from multiple CAs, the subject key identifier provides a means to quickly identify the set of certificates containing a particular public key. To assist applications in identificiation the appropriate end entity certificate, this extension SHOULD be included in all end entity certificates.

For end entity certificates, subject key identifiers SHOULD be derived from the public key. Two common methods for generating key identifiers from the public key are identifed above.

Where a key identifier has not been previously established, this specification recommends use of one of these methods for generating keyIdentifiers.

This extension MUST NOT be marked critical.

```
id-ce-subjectKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 14 }

SubjectKeyIdentifier ::= KeyIdentifier
```

4.2.1.3 Key Usage

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

```
   id-ce-keyUsage OBJECT IDENTIFIER ::=  { id-ce 15 }

   KeyUsage ::= BIT STRING {
        digitalSignature        (0),
        nonRepudiation          (1),
        keyEncipherment         (2),
        dataEncipherment        (3),
        keyAgreement            (4),
        keyCertSign             (5),
```

```
        cRLSign                 (6),
        encipherOnly            (7),
        decipherOnly            (8) }
```

Bits in the KeyUsage type are used as follows:

The digitalSignature bit is asserted when the subject public key is used with a digital signature mechanism to support security services other than 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.

The nonRepudiation bit is asserted when the subject public key is used to verify digital signatures used to provide a non-repudiation service which protects against the signing entity falsely denying some action, excluding certificate or CRL signing.

The keyEncipherment bit is asserted when the subject public key is used for key transport. For example, when an RSA key is to be used for key management, then this bit shall asserted.

The dataEncipherment bit is asserted when the subject public key is used for enciphering user data, other than cryptographic keys.

The keyAgreement bit is asserted when the subject public key is used for key agreement. For example, when a Diffie-Hellman key is to be used for key management, then this bit shall asserted.

The keyCertSign bit is asserted when the subject public key is used for verifying a signature on certificates. This bit may only be asserted in CA certificates.

The cRLSign bit is asserted when the subject public key is used for verifying a signature on revocation information (e.g., a CRL).

The meaning of the encipherOnly bit is undefined in the absence of the keyAgreement bit. When the encipherOnly bit is asserted and the keyAgreement bit is also set, the subject public key may be used only for enciphering data while performing key agreement.

The meaning of the decipherOnly bit is undefined in the absence of the keyAgreement bit. When the decipherOnly bit is asserted and the keyAgreement bit is also set, the subject public key may be used only for deciphering data while performing key agreement.

This profile does not restrict the combinations of bits that may be set in an instantiation of the keyUsage extension. However, appropriate values for keyUsage extensions for particular algorithms are specified in section 7.3.

4.2.1.4 Private Key Usage Period

This profile recommends against the use of this extension. CAs conforming to this profile MUST NOT generate certificates with critical private key usage period extensions.

The private key usage period extension allows the certificate issuer to specify a different validity period for the private key than the certificate. This extension is intended for use with digital signature keys. This extension consists of two optional components, notBefore and notAfter. The private key associated with the certificate should not be used to sign objects before or after the times specified by the two components, respectively. CAs conforming to this profile MUST NOT generate certificates with private key usage period extensions unless at least one of the two components is present.

Where used, notBefore and notAfter are represented as GeneralizedTime and MUST be specified and interpreted as defined in section 4.1.2.5.2.

```
id-ce-privateKeyUsagePeriod OBJECT IDENTIFIER ::=  { id-ce 16 }

PrivateKeyUsagePeriod ::= SEQUENCE {
     notBefore       [0]     GeneralizedTime OPTIONAL,
     notAfter        [1]     GeneralizedTime OPTIONAL }
```

4.2.1.5 Certificate Policies

The certificate policies extension contains a sequence of one or more policy information terms, each of which consists of an object identifier (OID) and optional qualifiers. These policy information terms indicate the policy under which the certificate has been issued and the purposes for which the certificate may be used. Optional qualifiers, which may be present, are not expected to change the definition of the policy.

Applications with specific policy requirements are expected to have a list of those policies which they will accept and to compare the policy OIDs in the certificate to that list. If this extension is critical, the path validation software MUST be able to interpret this extension (including the optional qualifier), or MUST reject the certificate.

To promote interoperability, this profile RECOMMENDS that policy information terms consist of only an OID. Where an OID alone is insufficient, this profile strongly recommends that use of qualifiers be limited to those identified in this section.

This specification defines two policy qualifier types for use by certificate policy writers and certificate issuers. The qualifier types are the CPS Pointer and User Notice qualifiers.

The CPS Pointer qualifier contains a pointer to a Certification Practice Statement (CPS) published by the CA. The pointer is in the form of a URI.

User notice is intended for display to a relying party when a certificate is used. The application software SHOULD display all user notices in all certificates of the certification path used, except that if a notice is duplicated only one copy need be displayed. To prevent such duplication, this qualifier SHOULD only be present in end-entity certificates and CA certificates issued to other organizations.

The user notice has two optional fields: the noticeRef field and the explicitText field.

The noticeRef field, if used, names an organization and identifies, by number, a particular textual statement prepared by that organization. For example, it might identify the organization "CertsRUs" and notice number 1. In a typical implementation, the application software will have a notice file containing the current set of notices for CertsRUs; the application will extract the notice text from the file and display it. Messages may be multilingual, allowing the software to select the particular language message for its own environment.

An explicitText field includes the textual statement directly in the certificate. The explicitText field is a string with a maximum size of 200 characters.

If both the noticeRef and explicitText options are included in the one qualifier and if the application software can locate the notice text indicated by the noticeRef option then that text should be displayed; otherwise, the explicitText string should be displayed.

```
id-ce-certificatePolicies OBJECT IDENTIFIER ::=  { id-ce 32 }

certificatePolicies ::= SEQUENCE SIZE (1..MAX) OF PolicyInformation
```

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

   CertPolicyId ::= OBJECT IDENTIFIER

   PolicyQualifierInfo ::= SEQUENCE {
        policyQualifierId  PolicyQualifierId,
        qualifier          ANY DEFINED BY policyQualifierId }

   -- policyQualifierIds for Internet policy qualifiers

   id-qt          OBJECT IDENTIFIER ::=  { id-pkix 2 }
   id-qt-cps      OBJECT IDENTIFIER ::=  { id-qt 1 }
   id-qt-unotice  OBJECT IDENTIFIER ::=  { id-qt 2 }

   PolicyQualifierId ::=
        OBJECT IDENTIFIER ( id-qt-cps | id-qt-unotice )

   Qualifier ::= CHOICE {
        cPSuri           CPSuri,
        userNotice       UserNotice }

   CPSuri ::= IA5String

   UserNotice ::= SEQUENCE {
        noticeRef        NoticeReference OPTIONAL,
        explicitText     DisplayText OPTIONAL}

   NoticeReference ::= SEQUENCE {
        organization     DisplayText,
        noticeNumbers    SEQUENCE OF INTEGER }

   DisplayText ::= CHOICE {
        visibleString    VisibleString  (SIZE (1..200)),
        bmpString        BMPString      (SIZE (1..200)),
        utf8String       UTF8String     (SIZE (1..200)) }
```

4.2.1.6 Policy Mappings

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

The issuing CA's users may accept an issuerDomainPolicy for certain applications. The policy mapping tells the issuing CA's users which policies associated with the subject CA are comparable to the policy they accept.

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

```
id-ce-policyMappings OBJECT IDENTIFIER ::=  { id-ce 33 }

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

4.2.1.7 Subject Alternative Name

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

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

Further, if the only subject identity included in the certificate is an alternative name form (e.g., an electronic mail address), then the subject distinguished name MUST be empty (an empty sequence), and the subjectAltName extension MUST be present. If the subject field contains an empty sequence, the subjectAltName extension MUST be marked critical.

When the subjectAltName extension contains an Internet mail address, the address MUST be included as an rfc822Name. The format of an rfc822Name is an "addr-spec" as defined in RFC 822 [RFC 822]. An addr-spec has the form "local-part@domain". Note that an addr-spec has no phrase (such as a common name) before it, has no comment (text surrounded in parentheses) after it, and is not surrounded by "<" and ">". Note that while upper and lower case letters are allowed in an RFC 822 addr-spec, no significance is attached to the case.

When the subjectAltName extension contains a iPAddress, the address MUST be stored in the octet string in "network byte order," as specified in RFC 791 [RFC 791]. The least significant bit (LSB) of

each octet is the LSB of the corresponding byte in the network address. For IP Version 4, as specified in RFC 791, the octet string MUST contain exactly four octets. For IP Version 6, as specified in RFC 1883, the octet string MUST contain exactly sixteen octets [RFC 1883].

When the subjectAltName extension contains a domain name service label, the domain name MUST be stored in the dNSName (an IA5String). The name MUST be in the "preferred name syntax," as specified by RFC 1034 [RFC 1034]. Note that while upper and lower case letters are allowed in domain names, no signifigance is attached to the case. In addition, while the string " " is a legal domain name, subjectAltName extensions with a dNSName " " are not permitted. Finally, the use of the DNS representation for Internet mail addresses (wpolk.nist.gov instead of wpolk@nist.gov) is not permitted; such identities are to be encoded as rfc822Name.

When the subjectAltName extension contains a URI, the name MUST be stored in the uniformResourceIdentifier (an IA5String). The name MUST be a non-relative URL, and MUST follow the URL syntax and encoding rules specified in [RFC 1738]. The name must include both a scheme (e.g., "http" or "ftp") and a scheme-specific-part. The scheme-specific-part must include a fully qualified domain name or IP address as the host.

As specified in [RFC 1738], the scheme name is not case-sensitive (e.g., "http" is equivalent to "HTTP"). The host part is also not case-sensitive, but other components of the scheme-specific-part may be case-sensitive. When comparing URIs, conforming implementations MUST compare the scheme and host without regard to case, but assume the remainder of the scheme-specific-part is case sensitive.

Subject alternative names may be constrained in the same manner as subject distinguished names using the name constraints extension as described in section 4.2.1.11.

If the subjectAltName extension is present, the sequence MUST contain at least one entry. Unlike the subject field, conforming CAs MUST NOT issue certificates with subjectAltNames containing empty GeneralName fields. For example, an rfc822Name is represented as an IA5String. While an empty string is a valid IA5String, such an rfc822Name is not permitted by this profile. The behavior of clients that encounter such a certificate when processing a certificication path is not defined by this profile.

Finally, the semantics of subject alternative names that include wildcard characters (e.g., as a placeholder for a set of names) are not addressed by this specification. Applications with specific requirements may use such names but shall define the semantics.

```
id-ce-subjectAltName OBJECT IDENTIFIER ::=  { id-ce 17 }

SubjectAltName ::= GeneralNames

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

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

OtherName ::= SEQUENCE {
     type-id    OBJECT IDENTIFIER,
     value      [0] EXPLICIT ANY DEFINED BY type-id }

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

4.2.1.8 Issuer Alternative Names

As with 4.2.1.7, this extension is used to associate Internet style identities with the certificate issuer. Issuer alternative names MUST be encoded as in 4.2.1.7.

Where present, this extension SHOULD NOT be marked critical.

```
id-ce-issuerAltName OBJECT IDENTIFIER ::=  { id-ce 18 }

IssuerAltName ::= GeneralNames
```

4.2.1.9 Subject Directory Attributes

The subject directory attributes extension is not recommended as an essential part of this profile, but it may be used in local environments. This extension MUST be non-critical.

```
id-ce-subjectDirectoryAttributes OBJECT IDENTIFIER ::=  { id-ce 9 }

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

4.2.1.10 Basic Constraints

The basic constraints extension identifies whether the subject of the certificate is a CA and how deep a certification path may exist through that CA.

The pathLenConstraint field is meaningful only if cA is set to TRUE. In this case, it gives the maximum number of CA certificates that may follow this certificate in a certification path. A value of zero indicates that only an end-entity certificate may follow in the path. Where it appears, the pathLenConstraint field MUST be greater than or equal to zero. Where pathLenConstraint does not appear, there is no limit to the allowed length of the certification path.

This extension MUST appear as a critical extension in all CA certificates. This extension SHOULD NOT appear in end entity certificates.

```
id-ce-basicConstraints OBJECT IDENTIFIER ::=  { id-ce 19 }

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

4.2.1.11 Name Constraints

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

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

Within this profile, the minimum and maximum fields are not used with any name forms, thus minimum is always zero, and maximum is always absent.

For URIs, the constraint applies to the host part of the name. The constraint may specify a host or a domain. Examples would be "foo.bar.com"; and ".xyz.com". When the the constraint begins with a period, it may be expanded with one or more subdomains. That is, the constraint ".xyz.com" is satisfied by both abc.xyz.com and abc.def.xyz.com. However, the constraint ".xyz.com" is not satisfied by "xyz.com". When the constraint does not begin with a period, it specifies a host.

A name constraint for Internat mail addresses may specify a particular mailbox, all addresses at a particular host, or all mailboxes in a domain. To indicate a particular mailbox, the constraint is the complete mail address. For example, "root@xyz.com" indicates the root mailbox on the host "xyz.com". To indicate all Internet mail addresses on a particular host, the constraint is specified as the host name. For example, the constraint "xyz.com" is satisfied by any mail address at the host "xyz.com". To specify any address within a domain, the constraint is specified with a leading period (as with URIs). For example, ".xyz.com" indicates all the Internet mail addresses in the domain "xyz.com", but Internet mail addresses on the host "xyz.com".

DNS name restrictions are expressed as foo.bar.com. Any subdomain satisfies the name constraint. For example, www.foo.bar.com would satisfy the constraint but bigfoo.bar.com would not.

Legacy implementations exist where an RFC 822 name is embedded in the subject distinguished name in an attribute of type EmailAddress (see sec. 4.1.2.6). When rfc822 names are constrained, but the certificate does not include a subject alternative name, the rfc822 name constraint MUST be applied to the attribute of type EmailAddress in the subject distinguished name. The ASN.1 syntax for EmailAddress and the corresponding OID are supplied in Appendix A and B.

Restrictions of the form directoryName MUST be applied to the subject field in the certificate and to the subjectAltName extensions of type directoryName. Restrictions of the form x400Address MUST be applied to subjectAltName extensions of type x400Address.

When applying restrictions of the form directoryName, an implementation MUST compare DN attributes. At a minimum, implementations MUST perform the DN comparison rules specified in Section 4.1.2.4. CAs issuing certificates with a restriction of the form directoryName SHOULD NOT rely on implementation of the full ISO DN name comparison algorithm. This implies name restrictions shall be stated identically to the encoding used in the subject field or subjectAltName extension.

The syntax of iPAddress MUST be as described in section 4.2.1.7 with the following additions specifically for Name Constraints. For IPv4 addresses, the ipAddress field of generalName MUST contain eight (8) octets, encoded in the style of RFC 1519 (CIDR) to represent an address range.[RFC 1519] For IPv6 addresses, the ipAddress field MUST contain 32 octets similarly encoded. For example, a name constraint for "class C" subnet 10.9.8.0 shall be represented as the octets 0A 09 08 00 FF FF FF 00, representing the CIDR notation 10.9.8.0/255.255.255.0.

The syntax and semantics for name constraints for otherName, ediPartyName, and registeredID are not defined by this specification.

```
id-ce-nameConstraints OBJECT IDENTIFIER ::=  { id-ce 30 }

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

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

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

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

4.2.1.12 Policy Constraints

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

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

If the requireExplicitPolicy field is present, subsequent certificates shall include an acceptable policy identifier. The value of requireExplicitPolicy indicates the number of additional certificates that may appear in the path before an explicit policy is required. An acceptable policy identifier is the identifier of a

policy required by the user of the certification path or the identifier of a policy which has been declared equivalent through policy mapping.

Conforming CAs MUST NOT issue certificates where policy constraints is a null sequence. That is, at least one of the inhibitPolicyMapping field or the requireExplicitPolicy field MUST be present. The behavior of clients that encounter a null policy constraints field is not addressed in this profile.

This extension may be critical or non-critical.

```
id-ce-policyConstraints OBJECT IDENTIFIER ::=  { id-ce 36 }

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

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

4.2.1.13 Extended key usage field

This field 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 field. This field is defined as follows:

```
id-ce-extKeyUsage OBJECT IDENTIFIER ::= {id-ce 37}

ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId

KeyPurposeId ::= OBJECT IDENTIFIER
```

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

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

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

If the extension is flagged non-critical, then it indicates the intended purpose or purposes of the key, and may be used in finding the correct key/certificate of an entity that has multiple keys/certificates. It is an advisory field and does not imply that usage of the key is restricted by the certification authority to the

purpose indicated. Certificate using applications may nevertheless require that a particular purpose be indicated in order for the certificate to be acceptable to that application.

If a certificate contains both a critical key usage field and a critical extended key usage field, then both fields MUST be processed independently and the certificate MUST only be used for a purpose consistent with both fields. If there is no purpose consistent with both fields, then the certificate MUST NOT be used for any purpose.

The following key usage purposes are defined by this profile:

```
id-kp OBJECT IDENTIFIER ::= { id-pkix 3 }

id-kp-serverAuth              OBJECT IDENTIFIER ::=   {id-kp 1}
-- TLS Web server authentication
-- Key usage bits that may be consistent: digitalSignature,
--                          keyEncipherment or keyAgreement
--
id-kp-clientAuth              OBJECT IDENTIFIER ::=   {id-kp 2}
-- TLS Web client authentication
-- Key usage bits that may be consistent: digitalSignature and/or
--                             keyAgreement
--
id-kp-codeSigning             OBJECT IDENTIFIER ::=   {id-kp 3}
-- Signing of downloadable executable code
-- Key usage bits that may be consistent: digitalSignature
--
id-kp-emailProtection         OBJECT IDENTIFIER ::=   {id-kp 4}
-- E-mail protection
-- Key usage bits that may be consistent: digitalSignature,
--                          nonRepudiation, and/or (keyEncipherment
--                          or keyAgreement)
--
id-kp-timeStamping    OBJECT IDENTIFIER ::= { id-kp 8 }
-- Binding the hash of an object to a time from an agreed-upon time
-- source. Key usage bits that may be consistent: digitalSignature,
--                          nonRepudiation
```

4.2.1.14 CRL Distribution Points

The CRL distribution points extension identifies how CRL information is obtained. The extension SHOULD be non-critical, but this profile recommends support for this extension by CAs and applications. Further discussion of CRL management is contained in section 5.

If the cRLDistributionPoints extension contains a DistributionPointName of type URI, the following semantics MUST be assumed: the URI is a pointer to the current CRL for the associated reasons and will be issued by the associated cRLIssuer. The expected values for the URI are those defined in 4.2.1.7. Processing rules for other values are not defined by this specification. If the distributionPoint omits reasons, the CRL MUST include revocations for all reasons. If the distributionPoint omits cRLIssuer, the CRL MUST be issued by the CA that issued the certificate.

```
id-ce-cRLDistributionPoints OBJECT IDENTIFIER ::=  { id-ce 31 }

cRLDistributionPoints ::= {
     CRLDistPointsSyntax }

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) }
```

4.2.2 Private Internet Extensions

This section defines one new extension for use in the Internet Public Key Infrastructure. This extension may be used to direct applications to identify an on-line validation service supporting the issuing CA. As the information may be available in multiple forms, each extension is a sequence of IA5String values, each of which represents a URI. The URI implicitly specifies the location and format of the information and the method for obtaining the information.

An object identifier is defined for the private extension. The object identifier associated with the private extension is defined under the arc id-pe within the id-pkix name space. Any future extensions defined for the Internet PKI will also be defined under the arc id-pe.

```
id-pkix  OBJECT IDENTIFIER  ::=
         { iso(1) identified-organization(3) dod(6) internet(1)
                 security(5) mechanisms(5) pkix(7) }

id-pe  OBJECT IDENTIFIER  ::=  { id-pkix 1 }
```

4.2.2.1 Authority Information Access

The authority information access extension indicates how to access CA information and services for the issuer of the certificate in which the extension appears. Information and services may include on-line validation services and CA policy data. (The location of CRLs is not specified in this extension; that information is provided by the cRLDistributionPoints extension.) This extension may be included in subject or CA certificates, and it MUST be non-critical.

```
id-pe-authorityInfoAccess OBJECT IDENTIFIER ::= { id-pe 1 }

AuthorityInfoAccessSyntax  ::=
        SEQUENCE SIZE (1..MAX) OF AccessDescription

AccessDescription  ::=  SEQUENCE {
        accessMethod          OBJECT IDENTIFIER,
        accessLocation        GeneralName  }

id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }

id-ad-caIssuers OBJECT IDENTIFIER ::= { id-ad 2 }
```

Each entry in the sequence AuthorityInfoAccessSyntax describes the format and location of additional information about the CA who issued the certificate in which this extension appears. The type and format of the information is specified by the accessMethod field; the accessLocation field specifies the location of the information. The retrieval mechanism may be implied by the accessMethod or specified by accessLocation.

This profile defines one OID for accessMethod. The id-ad-caIssuers OID is used when the additional information lists CAs that have issued certificates superior to the CA that issued the certificate

containing this extension. The referenced CA Issuers description is intended to aid certificate users in the selection of a certification path that terminates at a point trusted by the certificate user.

When id-ad-caIssuers appears as accessInfoType, the accessLocation field describes the referenced description server and the access protocol to obtain the referenced description. The accessLocation field is defined as a GeneralName, which can take several forms. Where the information is available via http, ftp, or ldap, accessLocation MUST be a uniformResourceIdentifier. Where the information is available via the directory access protocol (dap), accessLocation MUST be a directoryName. When the information is available via electronic mail, accessLocation MUST be an rfc822Name. The semantics of other name forms of accessLocation (when accessMethod is id-ad-caIssuers) are not defined by this specification.

Additional access descriptors may be defined in other PKIX specifications.

5 CRL and CRL Extensions Profile

As described above, one goal of this X.509 v2 CRL profile is to foster the creation of an interoperable and reusable Internet PKI. To achieve this goal, guidelines for the use of extensions are specified, and some assumptions are made about the nature of information included in the CRL.

CRLs may be used in a wide range of applications and environments covering a broad spectrum of interoperability goals and an even broader spectrum of operational and assurance requirements. This profile establishes a common baseline for generic applications requiring broad interoperability. The profile defines a baseline set of information that can be expected in every CRL. Also, the profile defines common locations within the CRL for frequently used attributes as well as common representations for these attributes.

This profile does not define any private Internet CRL extensions or CRL entry extensions.

Environments with additional or special purpose requirements may build on this profile or may replace it.

Conforming CAs are not required to issue CRLs if other revocation or certificate status mechanisms are provided. Conforming CAs that issue CRLs MUST issue version 2 CRLs, and CAs MUST include the date by which the next CRL will be issued in the nextUpdate field (see

sec. 5.1.2.5), the CRL number extension (see sec. 5.2.3) and the authority key identifier extension (see sec. 5.2.1). Conforming applications are required to process version 1 and 2 CRLs.

5.1 CRL Fields

The X.509 v2 CRL syntax is as follows. For signature calculation, the data that is to be signed is ASN.1 DER encoded. ASN.1 DER encoding is a tag, length, value encoding system for each element.

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

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

-- Version, Time, CertificateSerialNumber, and Extensions
-- are all defined in the ASN.1 in section 4.1

-- AlgorithmIdentifier is defined in section 4.1.1.2
```

The following items describe the use of the X.509 v2 CRL in the Internet PKI.

5.1.1 CertificateList Fields

The CertificateList is a SEQUENCE of three required fields. The fields are described in detail in the following subsections.

5.1.1.1 tbsCertList

The first field in the sequence is the tbsCertList. This field is itself a sequence containing the name of the issuer, issue date, issue date of the next list, the list of revoked certificates, and optional CRL extensions. Further, each entry on the revoked certificate list is defined by a sequence of user certificate serial number, revocation date, and optional CRL entry extensions.

5.1.1.2 signatureAlgorithm

The signatureAlgorithm field contains the algorithm identifier for the algorithm used by the CA to sign the CertificateList. The field is of type AlgorithmIdentifier, which is defined in section 4.1.1.2. Section 7.2 lists the supported algorithms for this specification. Conforming CAs MUST use the algorithm identifiers presented in section 7.2 when signing with a supported signature algorithm.

This field MUST contain the same algorithm identifier as the signature field in the sequence tbsCertList (see sec. 5.1.2.2).

5.1.1.3 signatureValue

The signatureValue field contains a digital signature computed upon the ASN.1 DER encoded tbsCertList. The ASN.1 DER encoded tbsCertList is used as the input to the signature function. This signature value is then ASN.1 encoded as a BIT STRING and included in the CRL's signatureValue field. The details of this process are specified for each of the supported algorithms in section 7.2.

5.1.2 Certificate List "To Be Signed"

The certificate list to be signed, or TBSCertList, is a SEQUENCE of required and optional fields. The required fields identify the CRL issuer, the algorithm used to sign the CRL, the date and time the CRL was issued, and the date and time by which the CA will issue the next CRL.

Optional fields include lists of revoked certificates and CRL extensions. The revoked certificate list is optional to support the case where a CA has not revoked any unexpired certificates that it has issued. The profile requires conforming CAs to use the CRL extension cRLNumber in all CRLs issued.

5.1.2.1 Version

This optional field describes the version of the encoded CRL. When extensions are used, as required by this profile, this field MUST be present and MUST specify version 2 (the integer value is 1).

5.1.2.2 Signature

This field contains the algorithm identifier for the algorithm used to sign the CRL. Section 7.2 lists OIDs for the most popular signature algorithms used in the Internet PKI.

This field MUST contain the same algorithm identifier as the signatureAlgorithm field in the sequence CertificateList (see section 5.1.1.2).

5.1.2.3 Issuer Name

The issuer name identifies the entity who has signed and issued the CRL. The issuer identity is carried in the issuer name field. Alternative name forms may also appear in the issuerAltName extension (see sec. 5.2.2). The issuer name field MUST contain an X.500 distinguished name (DN). The issuer name field is defined as the X.501 type Name, and MUST follow the encoding rules for the issuer name field in the certificate (see sec. 4.1.2.4).

5.1.2.4 This Update

This field indicates the issue date of this CRL. ThisUpdate may be encoded as UTCTime or GeneralizedTime.

CAs conforming to this profile that issue CRLs MUST encode thisUpdate as UTCTime for dates through the year 2049. CAs conforming to this profile that issue CRLs MUST encode thisUpdate as GeneralizedTime for dates in the year 2050 or later.

Where encoded as UTCTime, thisUpdate MUST be specified and interpreted as defined in section 4.1.2.5.1. Where encoded as GeneralizedTime, thisUpdate MUST be specified and interpreted as defined in section 4.1.2.5.2.

5.1.2.5 Next Update

This field indicates the date by which the next CRL will be issued. The next CRL could be issued before the indicated date, but it will not be issued any later than the indicated date. CAs SHOULD issue CRLs with a nextUpdate time equal to or later than all previous CRLs. nextUpdate may be encoded as UTCTime or GeneralizedTime.

This profile requires inclusion of nextUpdate in all CRLs issued by conforming CAs. Note that the ASN.1 syntax of TBSCertList describes this field as OPTIONAL, which is consistent with the ASN.1 structure defined in [X.509]. The behavior of clients processing CRLs which omit nextUpdate is not specified by this profile.

CAs conforming to this profile that issue CRLs MUST encode nextUpdate as UTCTime for dates through the year 2049. CAs conforming to this profile that issue CRLs MUST encode nextUpdate as GeneralizedTime for dates in the year 2050 or later.

Where encoded as UTCTime, nextUpdate MUST be specified and interpreted as defined in section 4.1.2.5.1. Where encoded as GeneralizedTime, nextUpdate MUST be specified and interpreted as defined in section 4.1.2.5.2.

5.1.2.6 Revoked Certificates

Revoked certificates are listed. The revoked certificates are named by their serial numbers. Certificates revoked by the CA are uniquely identified by the certificate serial number. The date on which the revocation occurred is specified. The time for revocationDate MUST be expressed as described in section 5.1.2.4. Additional information may be supplied in CRL entry extensions; CRL entry extensions are discussed in section 5.3.

5.1.2.7 Extensions

This field may only appear if the version is 2 (see sec. 5.1.2.1). If present, this field is a SEQUENCE of one or more CRL extensions. CRL extensions are discussed in section 5.2.

5.2 CRL Extensions

The extensions defined by ANSI X9 and ISO/IEC/ITU for X.509 v2 CRLs [X.509] [X9.55] provide methods for associating additional attributes with CRLs. The X.509 v2 CRL format also allows communities to define private extensions to carry information unique to those communities. Each extension in a CRL may be designated as critical or non-critical. A CRL validation MUST fail if it encounters a critical extension which it does not know how to process. However, an unrecognized non-critical extension may be ignored. The following subsections present those extensions used within Internet CRLs. Communities may elect to include extensions in CRLs which are not defined in this specification. However, caution should be exercised in adopting any critical extensions in CRLs which might be used in a general context.

Conforming CAs that issue CRLs are required to include the authority key identifier (see sec. 5.2.1) and the CRL number (see sec. 5.2.3) extensions in all CRLs issued.

5.2.1 Authority Key Identifier

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

Conforming CAs MUST use the key identifier method, and MUST include this extension in all CRLs issued.

The syntax for this CRL extension is defined in section 4.2.1.1.

5.2.2 Issuer Alternative Name

The issuer alternative names extension allows additional identities to be associated with the issuer of the CRL. Defined options include an rfc822 name (electronic mail address), a DNS name, an IP address, and a URI. Multiple instances of a name and multiple name forms may be included. Whenever such identities are used, the issuer alternative name extension MUST be used.

The issuerAltName extension SHOULD NOT be marked critical.

The OID and syntax for this CRL extension are defined in section 4.2.1.8.

5.2.3 CRL Number

The CRL number is a non-critical CRL extension which conveys a monotonically increasing sequence number for each CRL issued by a CA. This extension allows users to easily determine when a particular CRL supersedes another CRL. CAs conforming to this profile MUST include this extension in all CRLs.

```
id-ce-cRLNumber OBJECT IDENTIFIER ::= { id-ce 20 }

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

5.2.4 Delta CRL Indicator

The delta CRL indicator is a critical CRL extension that identifies a delta-CRL. The use of delta-CRLs can significantly improve processing time for applications which store revocation information in a format other than the CRL structure. This allows changes to be added to the local database while ignoring unchanged information that is already in the local database.

When a delta-CRL is issued, the CAs MUST also issue a complete CRL.

The value of BaseCRLNumber identifies the CRL number of the base CRL that was used as the starting point in the generation of this delta-CRL. The delta-CRL contains the changes between the base CRL and the current CRL issued along with the delta-CRL. It is the decision of a CA as to whether to provide delta-CRLs. Again, a delta-CRL MUST NOT be issued without a corresponding complete CRL. The value of CRLNumber for both the delta-CRL and the corresponding complete CRL MUST be identical.

A CRL user constructing a locally held CRL from delta-CRLs MUST consider the constructed CRL incomplete and unusable if the CRLNumber of the received delta-CRL is more than one greater than the CRLnumber of the delta-CRL last processed.

```
id-ce-deltaCRLIndicator OBJECT IDENTIFIER ::= { id-ce 27 }

deltaCRLIndicator ::= BaseCRLNumber

BaseCRLNumber ::= CRLNumber
```

5.2.5 Issuing Distribution Point

The issuing distribution point is a critical CRL extension that identifies the CRL distribution point for a particular CRL, and it indicates whether the CRL covers revocation for end entity certificates only, CA certificates only, or a limitied set of reason codes. Although the extension is critical, conforming implementations are not required to support this extension.

The CRL is signed using the CA’s private key. CRL Distribution Points do not have their own key pairs. If the CRL is stored in the X.500 Directory, it is stored in the Directory entry corresponding to the CRL distribution point, which may be different than the Directory entry of the CA.

The reason codes associated with a distribution point shall be specified in onlySomeReasons. If onlySomeReasons does not appear, the distribution point shall contain revocations for all reason codes. CAs may use CRL distribution points to partition the CRL on the basis of compromise and routine revocation. In this case, the revocations with reason code keyCompromise (1) and cACompromise (2) appear in one distribution point, and the revocations with other reason codes appear in another distribution point.

Where the issuingDistributionPoint extension contains a URL, the following semantics MUST be assumed: the object is a pointer to the most current CRL issued by this CA. The URI schemes ftp, http, mailto [RFC1738] and ldap [RFC1778] are defined for this purpose. The URI MUST be an absolute, not relative, pathname and MUST specify the host.

```
id-ce-issuingDistributionPoint OBJECT IDENTIFIER ::= { id-ce 28 }

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

5.3 CRL Entry Extensions

The CRL entry extensions already defined by ANSI X9 and ISO/IEC/ITU for X.509 v2 CRLs provide methods for associating additional attributes with CRL entries [X.509] [X9.55]. The X.509 v2 CRL format also allows communities to define private CRL entry extensions to carry information unique to those communities. Each extension in a CRL entry may be designated as critical or non-critical. A CRL validation MUST fail if it encounters a critical CRL entry extension which it does not know how to process. However, an unrecognized non-critical CRL entry extension may be ignored. The following subsections present recommended extensions used within Internet CRL entries and standard locations for information. Communities may elect to use additional CRL entry extensions; however, caution should be exercised in adopting any critical extensions in CRL entries which might be used in a general context.

All CRL entry extensions used in this specification are non-critical. Support for these extensions is optional for conforming CAs and applications. However, CAs that issue CRLs SHOULD include reason codes (see sec. 5.3.1) and invalidity dates (see sec. 5.3.3) whenever this information is available.

5.3.1 Reason Code

The reasonCode is a non-critical CRL entry extension that identifies the reason for the certificate revocation. CAs are strongly encouraged to include meaningful reason codes in CRL entries; however, the reason code CRL entry extension SHOULD be absent instead of using the unspecified (0) reasonCode value.

```
id-ce-cRLReason OBJECT IDENTIFIER ::= { id-ce 21 }

-- reasonCode ::= { CRLReason }

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

5.3.2 Hold Instruction Code

The hold instruction code is a non-critical CRL entry extension that provides a registered instruction identifier which indicates the action to be taken after encountering a certificate that has been placed on hold.

```
id-ce-holdInstructionCode OBJECT IDENTIFIER ::= { id-ce 23 }

holdInstructionCode ::= OBJECT IDENTIFIER
```

The following instruction codes have been defined. Conforming applications that process this extension MUST recognize the following instruction codes.

```
holdInstruction    OBJECT IDENTIFIER ::=
                  { iso(1) member-body(2) us(840) x9-57(10040) 2 }

id-holdinstruction-none   OBJECT IDENTIFIER ::= {holdInstruction 1}
id-holdinstruction-callissuer
                          OBJECT IDENTIFIER ::= {holdInstruction 2}
id-holdinstruction-reject OBJECT IDENTIFIER ::= {holdInstruction 3}
```

Conforming applications which encounter an id-holdinstruction-callissuer MUST call the certificate issuer or reject the certificate. Conforming applications which encounter an id-

holdinstruction-reject MUST reject the certificate. The hold instruction id-holdinstruction-none is semantically equivalent to the absence of a holdInstructionCode, and its use is strongly deprecated for the Internet PKI.

5.3.3 Invalidity Date

The invalidity date is a non-critical CRL entry extension that provides the date on which it is known or suspected that the private key was compromised or that the certificate otherwise became 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. When a revocation is first posted by a CA in a CRL, the invalidity date may precede the date of issue of earlier CRLs, but the revocation date SHOULD NOT precede the date of issue of earlier CRLs. Whenever this information is available, CAs are strongly encouraged to share it with CRL users.

The GeneralizedTime values included in this field MUST be expressed in Greenwich Mean Time (Zulu), and MUST be specified and interpreted as defined in section 4.1.2.5.2.

```
id-ce-invalidityDate OBJECT IDENTIFIER ::= { id-ce 24 }

invalidityDate ::=  GeneralizedTime
```

5.3.4 Certificate Issuer

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

```
id-ce-certificateIssuer   OBJECT IDENTIFIER ::= { id-ce 29 }

certificateIssuer ::=     GeneralNames
```

If used by conforming CAs that issue CRLs, this extension is always critical. If an implementation ignored this extension it could not correctly attribute CRL entries to certificates. This specification RECOMMENDS that implementations recognize this extension.

6 Certification Path Validation

Certification path validation procedures for the Internet PKI are based on section 12.4.3 of [X.509]. Certification path processing verifies the binding between the subject distinguished name and/or subject alternative name and subject public key. The binding is limited by constraints which are specified in the certificates which comprise the path. The basic constraints and policy constraints extensions allow the certification path processing logic to automate the decision making process.

This section describes an algorithm for validating certification paths. Conforming implementations of this specification are not required to implement this algorithm, but MUST be functionally equivalent to the external behavior resulting from this procedure. Any algorithm may be used by a particular implementation so long as it derives the correct result.

In section 6.1, the text describes basic path validation. This text assumes that all valid paths begin with certificates issued by a single "most-trusted CA". The algorithm requires the public key of the CA, the CA's name, the validity period of the public key, and any constraints upon the set of paths which may be validated using this key.

The "most-trusted CA" is a matter of policy: it could be a root CA in a hierarchical PKI; the CA that issued the verifier's own certificate(s); or any other CA in a network PKI. The path validation procedure is the same regardless of the choice of "most-trusted CA."

section 6.2 describes extensions to the basic path validation algorithm. Two specific cases are discussed: the case where paths may begin with one of several trusted CAs; and where compatibility with the PEM architecture is required.

6.1 Basic Path Validation

The text assumes that the trusted public key (and related information) is contained in a "self-signed" certificate. This simplifies the description of the path processing procedure. Note that the signature on the self-signed certificate does not provide any security services. The trusted public key (and related information) may be obtained in other formats; the information is trusted because of other procedures used to obtain and protect it.

The goal of path validation is to verify the binding between a subject distinguished name or subject alternative name and subject public key, as represented in the "end entity" certificate, based on the public key of the "most-trusted CA". This requires obtaining a sequence of certificates that support that binding. The procedures performed to obtain this sequence is outside the scope of this section.

The following text also assumes that certificates do not use subject or unique identifier fields or private critical extensions, as recommended within this profile. However, if these components appear in certificates, they MUST be processed. Finally, policy qualifiers are also neglected for the sake of clarity.

A certification path is a sequence of n certificates where:

* for all x in {1,(n-1)}, the subject of certificate x is the issuer of certificate x+1.
* certificate x=1 is the the self-signed certificate, and
* certificate x=n is the end entity certificate.

This section assumes the following inputs are provided to the path processing logic:

(a) a certification path of length n;

(b) a set of initial policy identifiers (each comprising a sequence of policy element identifiers), which identifies one or more certificate policies, any one of which would be acceptable for the purposes of certification path processing, or the special value "any-policy";

(c) the current date/time (if not available internally to the certification path processing module); and

(d) the time, T, for which the validity of the path should be determined. (This may be the current date/time, or some point in the past.)

From the inputs, the procedure intializes five state variables:

(a) acceptable policy set: A set of certificate policy identifiers comprising the policy or policies recognized by the public key user together with policies deemed equivalent through policy mapping. The initial value of the acceptable policy set is the special value "any-policy".

(b) constrained subtrees: A set of root names defining a set of subtrees within which all subject names in subsequent certificates in the certification path shall fall. The initial value is "unbounded".

(c) excluded subtrees: A set of root names defining a set of subtrees within which no subject name in subsequent certificates in the certification path may fall. The initial value is "empty".

(d) explicit policy: an integer which indicates if an explicit policy identifier is required. The integer indicates the first certificate in the path where this requirement is imposed. Once set, this variable may be decreased, but may not be increased. (That is, if a certificate in the path requires explicit policy identifiers, a later certificate can not remove this requirement.) The initial value is n+1.

(e) policy mapping: an integer which indicates if policy mapping is permitted. The integer indicates the last certificate on which policy mapping may be applied. Once set, this variable may be decreased, but may not be increased. (That is, if a certificate in the path specifies policy mapping is not permitted, it can not be overriden by a later certificate.) The initial value is n+1.

The actions performed by the path processing software for each certificate i=1 through n are described below. The self-signed certificate is certificate i=1, the end entity certificate is i=n. The processing is performed sequentially, so that processing certificate i affects the state variables for processing certificate (i+1). Note that actions (h) through (m) are not applied to the end entity certificate (certificate n).

The path processing actions to be performed are:

(a) Verify the basic certificate information, including:

(1) the certificate was signed using the subject public key from certificate i-1 (in the special case i=1, this step may be omitted; if not, use the subject public key from the same certificate),

(2) the certificate validity period includes time T,

(3) the certificate had not been revoked at time T and is not currently on hold status that commenced before time T, (this may be determined by obtaining the appropriate CRL or status information, or by out-of-band mechanisms), and

(4) the subject and issuer names chain correctly (that is, the issuer of this certificate was the subject of the previous certificate.)

(b) Verify that the subject name and subjectAltName extension (critical or noncritical) is consistent with the constrained subtrees state variables.

(c) Verify that the subject name and subjectAltName extension (critical or noncritical) is consistent with the excluded subtrees state variables.

(d) Verify that policy information is consistent with the initial policy set:

(1) if the explicit policy state variable is less than or equal to i, a policy identifier in the certificate shall be in the initial policy set; and

(2) if the policy mapping variable is less than or equal to i, the policy identifier may not be mapped.

(e) Verify that policy information is consistent with the acceptable policy set:

(1) if the certificate policies extension is marked critical, the intersection of the policies extension and the acceptable policy set shall be non-null;

(2) the acceptable policy set is assigned the resulting intersection as its new value.

(g) Verify that the intersection of the acceptable policy set and the initial policy set is non-null.

(h) Recognize and process any other critical extension present in the certificate.

(i) Verify that the certificate is a CA certificate (as specified in a basicConstraints extension or as verified out-of-band).

(j) If permittedSubtrees is present in the certificate, set the constrained subtrees state variable to the intersection of its previous value and the value indicated in the extension field.

(k) If excludedSubtrees is present in the certificate, set the excluded subtrees state variable to the union of its previous value and the value indicated in the extension field.

(l) If a policy constraints extension is included in the certificate, modify the explicit policy and policy mapping state variables as follows:

(1) If requireExplicitPolicy is present and has value r, the explicit policy state variable is set to the minimum of its current value and the sum of r and i (the current certificate in the sequence).

(2) If inhibitPolicyMapping is present and has value q, the policy mapping state variable is set to the minimum of its current value and the sum of q and i (the current certificate in the sequence).

(m) If a key usage extension is marked critical, ensure the keyCertSign bit is set.

If any one of the above checks fail, the procedure terminates, returning a failure indication and an appropriate reason. If none of the above checks fail on the end-entity certificate, the procedure terminates, returning a success indication together with the set of all policy qualifier values encountered in the set of certificates.

6.2 Extending Path Validation

The path validation algorithm presented in 6.1 is based on several simplifying assumptions (e.g., a single trusted CA that starts all valid paths). This algorithm may be extended for cases where the assumptions do not hold.

This procedure may be extended for multiple trusted CAs by providing a set of self-signed certificates to the validation module. In this case, a valid path could begin with any one of the self-signed certificates. Limitations in the trust paths for any particular key may be incorporated into the self-signed certificate’s extensions. In this way, the self-signed certificates permit the path validation module to automatically incorporate local security policy and requirements.

It is also possible to specify an extended version of the above certification path processing procedure which results in default behavior identical to the rules of PEM [RFC 1422]. In this extended version, additional inputs to the procedure are a list of one or more Policy Certification Authorities (PCAs) names and an indicator of the position in the certification path where the PCA is expected. At the nominated PCA position, the CA name is compared against this list. If a recognized PCA name is found, then a constraint of SubordinateToCA is implicitly assumed for the remainder of the

certification path and processing continues. If no valid PCA name is found, and if the certification path cannot be validated on the basis of identified policies, then the certification path is considered invalid.

7 Algorithm Support

This section describes cryptographic algorithms which may be used with this profile. The section describes one-way hash functions and digital signature algorithms which may be used to sign certificates and CRLs, and identifies OIDs for public keys contained in a certificate.

Conforming CAs and applications are not required to support the algorithms or algorithm identifiers described in this section. However, conforming CAs and applications that use the algorithms identified here MUST support them as specified.

7.1 One-way Hash Functions

This section identifies one-way hash functions for use in the Internet PKI. One-way hash functions are also called message digest algorithms. SHA-1 is the preferred one-way hash function for the Internet PKI. However, PEM uses MD2 for certificates [RFC 1422] [RFC 1423] and MD5 is used in other legacy applications. For this reason, MD2 and MD5 are included in this profile.

7.1.1 MD2 One-way Hash Function

MD2 was developed by Ron Rivest for RSA Data Security. RSA Data Security has not placed the MD2 algorithm in the public domain. Rather, RSA Data Security has granted license to use MD2 for non-commercial Internet Privacy-Enhanced Mail. For this reason, MD2 may continue to be used with PEM certificates, but SHA-1 is preferred. MD2 produces a 128-bit "hash" of the input. MD2 is fully described in RFC 1319 [RFC 1319].

At the Selected Areas in Cryptography ’95 conference in May 1995, Rogier and Chauvaud presented an attack on MD2 that can nearly find collisions [RC95]. Collisions occur when one can find two different messages that generate the same message digest. A checksum operation in MD2 is the only remaining obstacle to the success of the attack. For this reason, the use of MD2 for new applications is discouraged. It is still reasonable to use MD2 to verify existing signatures, as the ability to find collisions in MD2 does not enable an attacker to find new messages having a previously computed hash value.

7.1.2 MD5 One-way Hash Function

MD5 was developed by Ron Rivest for RSA Data Security. RSA Data Security has placed the MD5 algorithm in the public domain. MD5 produces a 128-bit "hash" of the input. MD5 is fully described in RFC 1321 [RFC 1321].

Den Boer and Bosselaers [DB94] have found pseudo-collisions for MD5, but there are no other known cryptanalytic results. The use of MD5 for new applications is discouraged. It is still reasonable to use MD5 to verify existing signatures.

7.1.3 SHA-1 One-way Hash Function

SHA-1 was developed by the U.S. Government. SHA-1 produces a 160-bit "hash" of the input. SHA-1 is fully described in FIPS 180-1 [FIPS 180-1].

SHA-1 is the one-way hash function of choice for use with both the RSA and DSA signature algorithms (see sec. 7.2).

7.2 Signature Algorithms

Certificates and CRLs described by this standard may be signed with any public key signature algorithm. The certificate or CRL indicates the algorithm through an algorithm identifier which appears in the signatureAlgorithm field in a Certificate or CertificateList. This algorithm identifier is an OID and has optionally associated parameters. This section identifies algorithm identifiers and parameters that shall be used in the signatureAlgorithm field in a Certificate or CertificateList.

RSA and DSA are the most popular signature algorithms used in the Internet. Signature algorithms are always used in conjunction with a one-way hash function identified in section 7.1.

The signature algorithm and one-way hash function used to sign a certificate or CRL is indicated by use of an algorithm identifier. An algorithm identifier is an OID, and may include associated parameters. This section identifies OIDS for RSA and DSA. The contents of the parameters component for each algorithm vary; details are provided for each algorithm.

The data to be signed (e.g., the one-way hash function output value) is formatted for the signature algorithm to be used. Then, a private key operation (e.g., RSA encryption) is performed to generate the

signature value. This signature value is then ASN.1 encoded as a BIT STRING and included in the Certificate or CertificateList in the signature field.

7.2.1 RSA Signature Algorithm

A patent statement regarding the RSA algorithm can be found at the end of this profile.

The RSA algorithm is named for its inventors: Rivest, Shamir, and Adleman. This profile includes three signature algorithms based on the RSA asymmetric encryption algorithm. The signature algorithms combine RSA with either the MD2, MD5, or the SHA-1 one-way hash functions.

The signature algorithm with MD2 and the RSA encryption algorithm is defined in PKCS #1 [RFC 2313]. As defined in RFC 2313, the ASN.1 OID used to identify this signature algorithm is:

```
md2WithRSAEncryption OBJECT IDENTIFIER  ::=  {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
    pkcs-1(1) 2  }
```

The signature algorithm with MD5 and the RSA encryption algorithm is defined in PKCS #1 [RFC 2313]. As defined in RFC 2313, the ASN.1 OID used to identify this signature algorithm is:

```
md5WithRSAEncryption OBJECT IDENTIFIER  ::=  {
    iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
    pkcs-1(1) 4  }
```

The signature algorithm with SHA-1 and the RSA encryption algorithm is implemented using the padding and encoding conventions described in PKCS #1 [RFC 2313]. The message digest is computed using the SHA-1 hash algorithm. The ASN.1 object identifier used to identify this signature algorithm is:

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

When any of these three OIDs appears within the ASN.1 type AlgorithmIdentifier, the parameters component of that type shall be the ASN.1 type NULL.

The RSA signature generation process and the encoding of the result is described in detail in RFC 2313.

7.2.2 DSA Signature Algorithm

A patent statement regarding the DSA can be found at the end of this profile.

The Digital Signature Algorithm (DSA) is also called the Digital Signature Standard (DSS). DSA was developed by the U.S. Government, and DSA is used in conjunction with the the SHA-1 one-way hash function. DSA is fully described in FIPS 186 [FIPS 186]. The ASN.1 OIDs used to identify this signature algorithm are:

```
id-dsa-with-sha1 ID  ::=  {
        iso(1) member-body(2) us(840) x9-57 (10040)
        x9cm(4) 3 }
```

Where the id-dsa-with-sha1 algorithm identifier appears as the algorithm field in an AlgorithmIdentifier, the encoding shall omit the parameters field. That is, the AlgorithmIdentifier shall be a SEQUENCE of one component - the OBJECT IDENTIFIER id-dsa-with-sha1.

The DSA parameters in the subjectPublicKeyInfo field of the certificate of the issuer shall apply to the verification of the signature.

When signing, the DSA algorithm generates two values. These values are commonly referred to as r and s. To easily transfer these two values as one signature, they shall be ASN.1 encoded using the following ASN.1 structure:

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

7.3 Subject Public Key Algorithms

Certificates described by this profile may convey a public key for any public key algorithm. The certificate indicates the algorithm through an algorithm identifier. This algorithm identifier is an OID and optionally associated parameters.

This section identifies preferred OIDs and parameters for the RSA, DSA, and Diffie-Hellman algorithms. Conforming CAs shall use the identified OIDs when issuing certificates containing public keys for these algorithms. Conforming applications supporting any of these algorithms shall, at a minimum, recognize the OID identified in this section.

7.3.1 RSA Keys

The OID rsaEncryption identifies RSA public keys.

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

rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1}
```

The rsaEncryption OID is intended to be used in the algorithm field of a value of type AlgorithmIdentifier. The parameters field shall have ASN.1 type NULL for this algorithm identifier.

The RSA public key shall be encoded using the ASN.1 type RSAPublicKey:

```
RSAPublicKey ::= SEQUENCE {
   modulus            INTEGER, -- n
   publicExponent     INTEGER  -- e -- }
```

where modulus is the modulus n, and publicExponent is the public exponent e. The DER encoded RSAPublicKey is the value of the BIT STRING subjectPublicKey.

This OID is used in public key certificates for both RSA signature keys and RSA encryption keys. The intended application for the key may be indicated in the key usage field (see sec. 4.2.1.3). The use of a single key for both signature and encryption purposes is not recommended, but is not forbidden.

If the keyUsage extension is present in an end entity certificate which conveys an RSA public key, any combination of the following values may be present: digitalSignature; nonRepudiation; keyEncipherment; and dataEncipherment. If the keyUsage extension is present in a CA certificate which conveys an RSA public key, any combination of the following values may be present: digitalSignature; nonRepudiation; keyEncipherment; dataEncipherment; keyCertSign; and cRLSign. However, this specification RECOMMENDS that if keyCertSign or cRLSign is present, both keyEncipherment and dataEncipherment should not be present.

7.3.2 Diffie-Hellman Key Exchange Key

The Diffie-Hellman OID supported by this profile is defined by ANSI X9.42 [X9.42].

```
dhpublicnumber OBJECT IDENTIFIER ::= { iso(1) member-body(2)
          us(840) ansi-x942(10046) number-type(2) 1 }
```

The dhpublicnumber OID is intended to be used in the algorithm field of a value of type AlgorithmIdentifier. The parameters field of that type, which has the algorithm-specific syntax ANY DEFINED BY algorithm, have the ASN.1 type DomainParameters for this algorithm.

```
DomainParameters ::= SEQUENCE {
      p       INTEGER, -- odd prime, p=jq +1
      g       INTEGER, -- generator, g
      q       INTEGER, -- factor of p-1
      j       INTEGER OPTIONAL, -- subgroup factor
      validationParms  ValidationParms OPTIONAL }

ValidationParms ::= SEQUENCE {
      seed             BIT STRING,
      pgenCounter      INTEGER }
```

The fields of type DomainParameters have the following meanings:

p identifies the prime p defining the Galois field;

g specifies the generator of the multiplicative subgroup of order g;

q specifies the prime factor of p-1;

j optionally specifies the value that satisfies the equation p=jq+1 to support the optional verification of group parameters;

seed optionally specifies the bit string parameter used as the seed for the system parameter generation process; and

pgenCounter optionally specifies the integer value output as part of the of the system parameter prime generation process.

If either of the parameter generation components (pgencounter or seed) is provided, the other shall be present as well.

The Diffie-Hellman public key shall be ASN.1 encoded as an INTEGER; this encoding shall be used as the contents (i.e., the value) of the subjectPublicKey component (a BIT STRING) of the subjectPublicKeyInfo data element.

```
DHPublicKey ::= INTEGER -- public key, y = g^x mod p
```

If the keyUsage extension is present in a certificate which conveys a DH public key, the following values may be present: keyAgreement; encipherOnly; and decipherOnly. At most one of encipherOnly and decipherOnly shall be asserted in keyUsage extension.

7.3.3 DSA Signature Keys

The Digital Signature Algorithm (DSA) is also known as the Digital Signature Standard (DSS). The DSA OID supported by this profile is

```
id-dsa ID ::= { iso(1) member-body(2) us(840) x9-57(10040)
          x9cm(4) 1 }
```

The id-dsa algorithm syntax includes optional parameters. These parameters are commonly referred to as p, q, and g. When omitted, the parameters component shall be omitted entirely. That is, the AlgorithmIdentifier shall be a SEQUENCE of one component - the OBJECT IDENTIFIER id-dsa.

If the DSA algorithm parameters are present in the subjectPublicKeyInfo AlgorithmIdentifier, the parameters are included using the following ASN.1 structure:

```
Dss-Parms  ::=  SEQUENCE  {
    p             INTEGER,
    q             INTEGER,
    g             INTEGER  }
```

If the DSA algorithm parameters are absent from the subjectPublicKeyInfo AlgorithmIdentifier and the CA signed the subject certificate using DSA, then the certificate issuer's DSA parameters apply to the subject's DSA key. If the DSA algorithm parameters are absent from the subjectPublicKeyInfo AlgorithmIdentifier and the CA signed the subject certificate using a signature algorithm other than DSA, then the subject's DSA parameters are distributed by other means. If the subjectPublicKeyInfo AlgorithmIdentifier field omits the parameters component and the CA signed the subject with a signature algorithm other than DSA, then clients shall reject the certificate.

When signing, DSA algorithm generates two values. These values are commonly referred to as r and s. To easily transfer these two values as one signature, they are ASN.1 encoded using the following ASN.1 structure:

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

The encoded signature is conveyed as the value of the BIT STRING signature in a Certificate or CertificateList.

The DSA public key shall be ASN.1 DER encoded as an INTEGER; this encoding shall be used as the contents (i.e., the value) of the subjectPublicKey component (a BIT STRING) of the SubjectPublicKeyInfo data element.

```
DSAPublicKey ::= INTEGER -- public key, Y
```

If the keyUsage extension is present in an end entity certificate which conveys a DSA public key, any combination of the following values may be present: digitalSignature; and nonRepudiation.

If the keyUsage extension is present in an CA certificate which conveys a DSA public key, any combination of the following values may be present: digitalSignature; nonRepudiation; keyCertSign; and cRLSign.

8 References

[FIPS 180-1] Federal Information Processing Standards Publication (FIPS PUB) 180-1, Secure Hash Standard, 17 April 1995. [Supersedes FIPS PUB 180 dated 11 May 1993.]

[FIPS 186] Federal Information Processing Standards Publication (FIPS PUB) 186, Digital Signature Standard, 18 May 1994.

[RC95] Rogier, N. and Chauvaud, P., "The compression function of MD2 is not collision free," Presented at Selected Areas in Cryptography ’95, May 1995.

[RFC 791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC 822] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[RFC 1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC 1319] Kaliski, B., "The MD2 Message-Digest Algorithm," RFC 1319, April 1992.

[RFC 1321] Rivest, R., "The MD5 Message-Digest Algorithm," RFC 1321, April 1992.

[RFC 1422] Kent, S., "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management," RFC 1422, February 1993.

[RFC 1423] Balenson, D., "Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers," RFC 1423, February 1993.

[RFC 1519] Fuller, V., Li, T., Yu, J. and K. Varadhan. "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", RFC 1519, September 1993.

[RFC 1738] Berners-Lee, T., Masinter L., and M. McCahill. "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[RFC 1778] Howes, T., Kille S., Yeong, W. and C. Robbins. "The String Representation of Standard Attribute Syntaxes," RFC 1778, March 1995.

[RFC 1883] Deering, S. and R. Hinden. "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, December 1995.

[RFC 2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC 2247] Kille, S., Wahl, M., Grimstad, A., Huber, R. and S. Sataluri. "Using Domains in LDAP/X.500 Distinguished Names", RFC 2247, January 1998.

[RFC 2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", RFC 2277, January 1998.

[RFC 2279] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[RFC 2313] Kaliski, B., "PKCS #1: RSA Encryption Version 1.5", RFC 2313, March 1998.

[SDN.701] SDN.701, "Message Security Protocol 4.0", Revision A 1997-02-06.

[X.208] CCITT Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1), 1988.

[X.501] ITU-T Recommendation X.501: Information Technology - Open Systems Interconnection - The Directory: Models, 1993.

[X.509] ITU-T Recommendation X.509 (1997 E): Information Technology - Open Systems Interconnection - The Directory: Authentication Framework, June 1997.

[X.520] ITU-T Recommendation X.520: Information Technology - Open Systems Interconnection - The Directory: Selected Attribute Types, 1993.

[X9.42] ANSI X9.42-199x, Public Key Cryptography for The Financial Services Industry: Agreement of Symmetric Algorithm Keys Using Diffie-Hellman (Working Draft), December 1997.

[X9.55] ANSI X9.55-1995, Public Key Cryptography For The Financial Services Industry: Extensions To Public Key Certificates And Certificate Revocation Lists, 8 December, 1995.

[X9.57] ANSI X9.57-199x, Public Key Cryptography For The Financial Services Industry: Certificate Management (Working Draft), 21 June, 1996.

9 Intellectual Property Rights

The IETF has been notified of intellectual property rights claimed in regard to some or all of the specification contained in this document. For more information consult the online list of claimed rights.

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to

obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

10 Security Considerations

The majority of this specification is devoted to the format and content of certificates and CRLs. Since certificates and CRLs are digitally signed, no additional integrity service is necessary. Neither certificates nor CRLs need be kept secret, and unrestricted and anonymous access to certificates and CRLs has no security implications.

However, security factors outside the scope of this specification will affect the assurance provided to certificate users. This section highlights critical issues that should be considered by implementors, administrators, and users.

The procedures performed by CAs and RAs to validate the binding of the subject's identity of their public key greatly affect the assurance that should be placed in the certificate. Relying parties may wish to review the CA's certificate practice statement. This may be particularly important when issuing certificates to other CAs.

The use of a single key pair for both signature and other purposes is strongly discouraged. Use of separate key pairs for signature and key management provides several benefits to the users. The ramifications associated with loss or disclosure of a signature key are different from loss or disclosure of a key management key. Using separate key pairs permits a balanced and flexible response. Similarly, different validity periods or key lengths for each key pair may be appropriate in some application environments. Unfortunately, some legacy applications (e.g., SSL) use a single key pair for signature and key management.

The protection afforded private keys is a critical factor in maintaining security. On a small scale, failure of users to protect their private keys will permit an attacker to masquerade as them, or decrypt their personal information. On a larger scale, compromise of a CA's private signing key may have a catastrophic effect. If an attacker obtains the private key unnoticed, the attacker may issue bogus certificates and CRLs. Existence of bogus certificates and CRLs will undermine confidence in the system. If the compromise is detected, all certificates issued to the CA shall be revoked, preventing services between its users and users of other CAs. Rebuilding after such a compromise will be problematic, so CAs are advised to implement a combination of strong technical measures (e.g., tamper-resistant cryptographic modules) and appropriate

management procedures (e.g., separation of duties) to avoid such an incident.

Loss of a CA's private signing key may also be problematic. The CA would not be able to produce CRLs or perform normal key rollover. CAs are advised to maintain secure backup for signing keys. The security of the key backup procedures is a critical factor in avoiding key compromise.

The availability and freshness of revocation information will affect the degree of assurance that should be placed in a certificate. While certificates expire naturally, events may occur during its natural lifetime which negate the binding between the subject and public key. If revocation information is untimely or unavailable, the assurance associated with the binding is clearly reduced. Similarly, implementations of the Path Validation mechanism described in section 6 that omit revocation checking provide less assurance than those that support it.

The path validation algorithm depends on the certain knowledge of the public keys (and other information) about one or more trusted CAs. The decision to trust a CA is an important decision as it ultimately determines the trust afforded a certificate. The authenticated distribution of trusted CA public keys (usually in the form of a "self-signed" certificate) is a security critical out of band process that is beyond the scope of this specification.

In addition, where a key compromise or CA failure occurs for a trusted CA, the user will need to modify the information provided to the path validation routine. Selection of too many trusted CAs will make the trusted CA information difficult to maintain. On the other hand, selection of only one trusted CA may limit users to a closed community of users until a global PKI emerges.

The quality of implementations that process certificates may also affect the degree of assurance provided. The path validation algorithm described in section 6 relies upon the integrity of the trusted CA information, and especially the integrity of the public keys associated with the trusted CAs. By substituting public keys for which an attacker has the private key, an attacker could trick the user into accepting false certificates.

The binding between a key and certificate subject cannot be stronger than the cryptographic module implementation and algorithms used to generate the signature. Short key lengths or weak hash algorithms will limit the utility of a certificate. CAs are encouraged to note advances in cryptology so they can employ strong cryptographic techniques. In addition, CAs should decline to issue certificates to

CAs or end entities that generate weak signatures.

Inconsistent application of name comparison rules may result in acceptance of invalid X.509 certification paths, or rejection of valid ones. The X.500 series of specifications defines rules for comparing distinguished names require comparison of strings without regard to case, character set, multi-character white space substring, or leading and trailing white space. This specification relaxes these requirements, requiring support for binary comparison at a minimum.

CAs shall encode the distinguished name in the subject field of a CA certificate identically to the distinguished name in the issuer field in certificates issued by the latter CA. If CAs use different encodings, implementations of this specification may fail to recognize name chains for paths that include this certificate. As a consequence, valid paths could be rejected.

In addition, name constraints for distinguished names shall be stated identically to the encoding used in the subject field or subjectAltName extension. If not, (1) name constraints stated as excludedSubTrees will not match and invalid paths will be accepted and (2) name constraints expressed as permittedSubtrees will not match and valid paths will be rejected. To avoid acceptance of invalid paths, CAs should state name constraints for distinguished names as permittedSubtrees where ever possible.

Appendix A. Psuedo-ASN.1 Structures and OIDs

This section describes data objects used by conforming PKI components in an "ASN.1-like" syntax. This syntax is a hybrid of the 1988 and 1993 ASN.1 syntaxes. The 1988 ASN.1 syntax is augmented with 1993 UNIVERSAL Types UniversalString, BMPString and UTF8String.

The ASN.1 syntax does not permit the inclusion of type statements in the ASN.1 module, and the 1993 ASN.1 standard does not permit use of the new UNIVERSAL types in modules using the 1988 syntax. As a result, this module does not conform to either version of the ASN.1 standard.

This appendix may be converted into 1988 ASN.1 by replacing the defintions for the UNIVERSAL Types with the 1988 catch-all "ANY".

A.1 Explicitly Tagged Module, 1988 Syntax

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


DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

-- IMPORTS NONE --

-- UNIVERSAL Types defined in '93 and '98 ASN.1
-- but required by this specification

UniversalString ::= [UNIVERSAL 28] IMPLICIT OCTET STRING
        -- UniversalString is defined in ASN.1:1993

BMPString ::= [UNIVERSAL 30] IMPLICIT OCTET STRING
      -- BMPString is the subtype of UniversalString and models
       -- the Basic Multilingual Plane of ISO/IEC/ITU 10646-1

UTF8String ::= [UNIVERSAL 12] IMPLICIT OCTET STRING
        -- The content of this type conforms to RFC 2279.

--
-- PKIX specific OIDs

id-pkix  OBJECT IDENTIFIER  ::=
         { iso(1) identified-organization(3) dod(6) internet(1)
```

```
                    security(5) mechanisms(5) pkix(7) }
-- PKIX arcs

id-pe OBJECT IDENTIFIER  ::=  { id-pkix 1 }
        -- arc for private certificate extensions
id-qt OBJECT IDENTIFIER ::= { id-pkix 2 }
        -- arc for policy qualifier types
id-kp OBJECT IDENTIFIER ::= { id-pkix 3 }
        -- arc for extended key purpose OIDS
id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }
        -- arc for access descriptors

-- policyQualifierIds for Internet policy qualifiers

id-qt-cps      OBJECT IDENTIFIER ::=  { id-qt 1 }
        -- OID for CPS qualifier
id-qt-unotice  OBJECT IDENTIFIER ::=  { id-qt 2 }
        -- OID for user notice qualifier

-- access descriptor definitions

id-ad-ocsp      OBJECT IDENTIFIER ::= { id-ad 1 }
id-ad-caIssuers OBJECT IDENTIFIER ::= { id-ad 2 }

-- attribute data types --

Attribute       ::=     SEQUENCE {
        type            AttributeType,
        values  SET OF AttributeValue
                -- at least one value is required -- }

AttributeType           ::=   OBJECT IDENTIFIER

AttributeValue          ::=   ANY

AttributeTypeAndValue           ::=     SEQUENCE {
        type    AttributeType,
        value   AttributeValue }

-- suggested naming attributes: Definition of the following
--  information object set may be augmented to meet local
--  requirements.  Note that deleting members of the set may
--  prevent interoperability with conforming implementations.
--  presented in pairs: the AttributeType followed by the
--  type definition for the corresponding AttributeValue

--Arc for standard naming attributes
id-at           OBJECT IDENTIFIER ::= {joint-iso-ccitt(2) ds(5) 4}
```

```
-- Attributes of type NameDirectoryString
id-at-name              AttributeType   ::=     {id-at 41}
id-at-surname           AttributeType   ::=     {id-at 4}
id-at-givenName         AttributeType   ::=     {id-at 42}
id-at-initials          AttributeType   ::=     {id-at 43}
id-at-generationQualifier       AttributeType   ::=     {id-at 44}

X520name        ::= CHOICE {
      teletexString         TeletexString (SIZE (1..ub-name)),
      printableString       PrintableString (SIZE (1..ub-name)),
      universalString       UniversalString (SIZE (1..ub-name)),
      utf8String            UTF8String (SIZE (1..ub-name)),
      bmpString             BMPString (SIZE(1..ub-name))   }

--

id-at-commonName        AttributeType   ::=     {id-at 3}

X520CommonName  ::=      CHOICE {
      teletexString         TeletexString (SIZE (1..ub-common-name)),
      printableString       PrintableString (SIZE (1..ub-common-name)),
      universalString       UniversalString (SIZE (1..ub-common-name)),
      utf8String            UTF8String (SIZE (1..ub-common-name)),
      bmpString             BMPString (SIZE(1..ub-common-name))   }

--

id-at-localityName      AttributeType   ::=     {id-at 7}

X520LocalityName ::= CHOICE {
      teletexString       TeletexString (SIZE (1..ub-locality-name)),
      printableString     PrintableString (SIZE (1..ub-locality-name)),
      universalString     UniversalString (SIZE (1..ub-locality-name)),
      utf8String          UTF8String (SIZE (1..ub-locality-name)),
      bmpString           BMPString (SIZE(1..ub-locality-name))   }

--

id-at-stateOrProvinceName       AttributeType   ::=     {id-at 8}

X520StateOrProvinceName         ::= CHOICE {
      teletexString       TeletexString (SIZE (1..ub-state-name)),
      printableString     PrintableString (SIZE (1..ub-state-name)),
      universalString     UniversalString (SIZE (1..ub-state-name)),
      utf8String          UTF8String (SIZE (1..ub-state-name)),
      bmpString           BMPString (SIZE(1..ub-state-name))   }

--
```

```
id-at-organizationName          AttributeType   ::=     {id-at 10}

X520OrganizationName ::= CHOICE {
  teletexString     TeletexString (SIZE (1..ub-organization-name)),
  printableString   PrintableString (SIZE (1..ub-organization-name)),
  universalString   UniversalString (SIZE (1..ub-organization-name)),
  utf8String        UTF8String (SIZE (1..ub-organization-name)),
  bmpString         BMPString (SIZE(1..ub-organization-name))   }

--

id-at-organizationalUnitName    AttributeType   ::=     {id-at 11}

X520OrganizationalUnitName ::= CHOICE {
 teletexString    TeletexString (SIZE (1..ub-organizational-unit-name)),
 printableString        PrintableString
                      (SIZE (1..ub-organizational-unit-name)),
 universalString        UniversalString
                      (SIZE (1..ub-organizational-unit-name)),
 utf8String       UTF8String (SIZE (1..ub-organizational-unit-name)),
 bmpString        BMPString (SIZE(1..ub-organizational-unit-name))   }

--

id-at-title     AttributeType   ::=     {id-at 12}

X520Title ::=   CHOICE {
      teletexString         TeletexString (SIZE (1..ub-title)),
      printableString       PrintableString (SIZE (1..ub-title)),
      universalString       UniversalString (SIZE (1..ub-title)),
      utf8String            UTF8String (SIZE (1..ub-title)),
      bmpString             BMPString (SIZE(1..ub-title))   }

--

id-at-dnQualifier       AttributeType   ::=     {id-at 46}
X520dnQualifier ::=     PrintableString

id-at-countryName       AttributeType   ::=     {id-at 6}
X520countryName ::=     PrintableString (SIZE (2)) -- IS 3166 codes


 -- Legacy attributes

pkcs-9 OBJECT IDENTIFIER ::=
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 9 }

emailAddress AttributeType      ::= { pkcs-9 1 }
```

```
Pkcs9email ::= IA5String (SIZE (1..ub-emailaddress-length))

-- naming data types --

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

RDNSequence     ::=   SEQUENCE OF RelativeDistinguishedName

DistinguishedName       ::=   RDNSequence

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

-- Directory string type --

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

-- certificate and CRL specific structures begin here

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

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

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

CertificateSerialNumber  ::=  INTEGER
```

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

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

UniqueIdentifier  ::=  BIT STRING

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

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

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

-- CRL structures

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

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

-- Version, Time, CertificateSerialNumber, and Extensions were
-- defined earlier for use in the certificate structure

AlgorithmIdentifier  ::=  SEQUENCE  {
```

```
     algorithm               OBJECT IDENTIFIER,
     parameters              ANY DEFINED BY algorithm OPTIONAL  }
                                -- contains a value of the type
                                -- registered for use with the
                                -- algorithm object identifier value

-- Algorithm OIDs and parameter structures

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

rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1 }

md2WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 2 }

md5WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 4 }

sha1WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 5 }

id-dsa-with-sha1 OBJECT IDENTIFIER ::=  {
     iso(1) member-body(2) us(840) x9-57 (10040) x9algorithm(4) 3 }

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

dhpublicnumber OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) ansi-x942(10046) number-type(2) 1 }

DomainParameters ::= SEQUENCE {
     p       INTEGER, -- odd prime, p=jq +1
     g       INTEGER, -- generator, g
     q       INTEGER, -- factor of p-1
     j       INTEGER OPTIONAL, -- subgroup factor, j>= 2
     validationParms  ValidationParms OPTIONAL }

ValidationParms ::= SEQUENCE {
     seed             BIT STRING,
     pgenCounter      INTEGER }

id-dsa OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) x9-57(10040) x9algorithm(4) 1 }

Dss-Parms  ::=  SEQUENCE  {
     p             INTEGER,
     q             INTEGER,
     g             INTEGER  }
```

```
-- x400 address syntax starts here
--      OR Names

ORAddress ::= SEQUENCE {
   built-in-standard-attributes BuiltInStandardAttributes,
   built-in-domain-defined-attributes
                        BuiltInDomainDefinedAttributes OPTIONAL,
   -- see also teletex-domain-defined-attributes
   extension-attributes ExtensionAttributes OPTIONAL }
--      The OR-address is semantically absent from the OR-name if the
--      built-in-standard-attribute sequence is empty and the
--      built-in-domain-defined-attributes and extension-attributes are
--      both omitted.

--      Built-in Standard Attributes

BuiltInStandardAttributes ::= SEQUENCE {
   country-name CountryName OPTIONAL,
   administration-domain-name AdministrationDomainName OPTIONAL,
   network-address      [0] NetworkAddress OPTIONAL,
   -- see also extended-network-address
   terminal-identifier  [1] TerminalIdentifier OPTIONAL,
   private-domain-name  [2] PrivateDomainName OPTIONAL,
   organization-name    [3] OrganizationName OPTIONAL,
   -- see also teletex-organization-name
   numeric-user-identifier      [4] NumericUserIdentifier OPTIONAL,
   personal-name        [5] PersonalName OPTIONAL,
   -- see also teletex-personal-name
   organizational-unit-names    [6] OrganizationalUnitNames OPTIONAL
   -- see also teletex-organizational-unit-names -- }

CountryName ::= [APPLICATION 1] CHOICE {
   x121-dcc-code NumericString
               (SIZE (ub-country-name-numeric-length)),
   iso-3166-alpha2-code PrintableString
               (SIZE (ub-country-name-alpha-length)) }

AdministrationDomainName ::= [APPLICATION 2] CHOICE {
   numeric NumericString (SIZE (0..ub-domain-name-length)),
   printable PrintableString (SIZE (0..ub-domain-name-length)) }

NetworkAddress ::= X121Address  -- see also extended-network-address

X121Address ::= NumericString (SIZE (1..ub-x121-address-length))

TerminalIdentifier ::= PrintableString (SIZE (1..ub-terminal-id-length))

PrivateDomainName ::= CHOICE {
```

```
   numeric NumericString (SIZE (1..ub-domain-name-length)),
   printable PrintableString (SIZE (1..ub-domain-name-length)) }

OrganizationName ::= PrintableString
                         (SIZE (1..ub-organization-name-length))
-- see also teletex-organization-name

NumericUserIdentifier ::= NumericString
                         (SIZE (1..ub-numeric-user-id-length))

PersonalName ::= SET {
   surname [0] PrintableString (SIZE (1..ub-surname-length)),
   given-name [1] PrintableString
                       (SIZE (1..ub-given-name-length)) OPTIONAL,
   initials [2] PrintableString (SIZE (1..ub-initials-length)) OPTIONAL,
   generation-qualifier [3] PrintableString
                (SIZE (1..ub-generation-qualifier-length)) OPTIONAL }
-- see also teletex-personal-name

OrganizationalUnitNames ::= SEQUENCE SIZE (1..ub-organizational-units)
                                       OF OrganizationalUnitName
-- see also teletex-organizational-unit-names

OrganizationalUnitName ::= PrintableString (SIZE
                       (1..ub-organizational-unit-name-length))

--      Built-in Domain-defined Attributes

BuiltInDomainDefinedAttributes ::= SEQUENCE SIZE
                                (1..ub-domain-defined-attributes) OF
                                BuiltInDomainDefinedAttribute

BuiltInDomainDefinedAttribute ::= SEQUENCE {
   type PrintableString (SIZE
                       (1..ub-domain-defined-attribute-type-length)),
   value PrintableString (SIZE
                       (1..ub-domain-defined-attribute-value-length))}

--      Extension Attributes

ExtensionAttributes ::= SET SIZE (1..ub-extension-attributes) OF
                       ExtensionAttribute

ExtensionAttribute ::=  SEQUENCE {
   extension-attribute-type [0] INTEGER (0..ub-extension-attributes),
   extension-attribute-value [1]
                       ANY DEFINED BY extension-attribute-type }
```

```
-- Extension types and attribute values
--

common-name INTEGER ::= 1

CommonName ::= PrintableString (SIZE (1..ub-common-name-length))

teletex-common-name INTEGER ::= 2

TeletexCommonName ::= TeletexString (SIZE (1..ub-common-name-length))

teletex-organization-name INTEGER ::= 3

TeletexOrganizationName ::=
               TeletexString (SIZE (1..ub-organization-name-length))

teletex-personal-name INTEGER ::= 4

TeletexPersonalName ::= SET {
   surname [0] TeletexString (SIZE (1..ub-surname-length)),
   given-name [1] TeletexString
               (SIZE (1..ub-given-name-length)) OPTIONAL,
   initials [2] TeletexString (SIZE (1..ub-initials-length)) OPTIONAL,
   generation-qualifier [3] TeletexString (SIZE
               (1..ub-generation-qualifier-length)) OPTIONAL }

teletex-organizational-unit-names INTEGER ::= 5

TeletexOrganizationalUnitNames ::= SEQUENCE SIZE
        (1..ub-organizational-units) OF TeletexOrganizationalUnitName

TeletexOrganizationalUnitName ::= TeletexString
                        (SIZE (1..ub-organizational-unit-name-length))

pds-name INTEGER ::= 7

PDSName ::= PrintableString (SIZE (1..ub-pds-name-length))

physical-delivery-country-name INTEGER ::= 8

PhysicalDeliveryCountryName ::= CHOICE {
   x121-dcc-code NumericString (SIZE (ub-country-name-numeric-length)),
   iso-3166-alpha2-code PrintableString
                        (SIZE (ub-country-name-alpha-length)) }

postal-code INTEGER ::= 9

PostalCode ::= CHOICE {
```

```
   numeric-code NumericString (SIZE (1..ub-postal-code-length)),
   printable-code PrintableString (SIZE (1..ub-postal-code-length)) }

physical-delivery-office-name INTEGER ::= 10

PhysicalDeliveryOfficeName ::= PDSParameter

physical-delivery-office-number INTEGER ::= 11

PhysicalDeliveryOfficeNumber ::= PDSParameter

extension-OR-address-components INTEGER ::= 12

ExtensionORAddressComponents ::= PDSParameter

physical-delivery-personal-name INTEGER ::= 13

PhysicalDeliveryPersonalName ::= PDSParameter

physical-delivery-organization-name INTEGER ::= 14

PhysicalDeliveryOrganizationName ::= PDSParameter

extension-physical-delivery-address-components INTEGER ::= 15

ExtensionPhysicalDeliveryAddressComponents ::= PDSParameter

unformatted-postal-address INTEGER ::= 16

UnformattedPostalAddress ::= SET {
   printable-address SEQUENCE SIZE (1..ub-pds-physical-address-lines) OF
           PrintableString (SIZE (1..ub-pds-parameter-length)) OPTIONAL,
   teletex-string TeletexString
         (SIZE (1..ub-unformatted-address-length)) OPTIONAL }

street-address INTEGER ::= 17

StreetAddress ::= PDSParameter

post-office-box-address INTEGER ::= 18

PostOfficeBoxAddress ::= PDSParameter

poste-restante-address INTEGER ::= 19

PosteRestanteAddress ::= PDSParameter

unique-postal-name INTEGER ::= 20
```

```
UniquePostalName ::= PDSParameter

local-postal-attributes INTEGER ::= 21

LocalPostalAttributes ::= PDSParameter

PDSParameter ::= SET {
   printable-string PrintableString
                (SIZE(1..ub-pds-parameter-length)) OPTIONAL,
   teletex-string TeletexString
                (SIZE(1..ub-pds-parameter-length)) OPTIONAL }

extended-network-address INTEGER ::= 22

ExtendedNetworkAddress ::= CHOICE {
   e163-4-address SEQUENCE {
        number [0] NumericString (SIZE (1..ub-e163-4-number-length)),
        sub-address [1] NumericString
                (SIZE (1..ub-e163-4-sub-address-length)) OPTIONAL },
   psap-address [0] PresentationAddress }

PresentationAddress ::= SEQUENCE {
        pSelector       [0] EXPLICIT OCTET STRING OPTIONAL,
        sSelector       [1] EXPLICIT OCTET STRING OPTIONAL,
        tSelector       [2] EXPLICIT OCTET STRING OPTIONAL,
        nAddresses      [3] EXPLICIT SET SIZE (1..MAX) OF OCTET STRING }

terminal-type  INTEGER ::= 23

TerminalType ::= INTEGER {
   telex (3),
   teletex (4),
   g3-facsimile (5),
   g4-facsimile (6),
   ia5-terminal (7),
   videotex (8) } (0..ub-integer-options)

--      Extension Domain-defined Attributes

teletex-domain-defined-attributes INTEGER ::= 6

TeletexDomainDefinedAttributes ::= SEQUENCE SIZE
   (1..ub-domain-defined-attributes) OF TeletexDomainDefinedAttribute

TeletexDomainDefinedAttribute ::= SEQUENCE {
        type TeletexString
               (SIZE (1..ub-domain-defined-attribute-type-length)),
        value TeletexString
```

```
               (SIZE (1..ub-domain-defined-attribute-value-length)) }

--  specifications of Upper Bounds shall be regarded as mandatory
--  from Annex B of ITU-T X.411 Reference Definition of MTS Parameter
--  Upper Bounds

--      Upper Bounds
ub-name INTEGER ::=     32768
ub-common-name  INTEGER ::=     64
ub-locality-name        INTEGER ::=     128
ub-state-name   INTEGER ::=     128
ub-organization-name    INTEGER ::=     64
ub-organizational-unit-name     INTEGER ::=     64
ub-title        INTEGER ::=     64
ub-match        INTEGER ::=     128

ub-emailaddress-length INTEGER ::= 128

ub-common-name-length INTEGER ::= 64
ub-country-name-alpha-length INTEGER ::= 2
ub-country-name-numeric-length INTEGER ::= 3
ub-domain-defined-attributes INTEGER ::= 4
ub-domain-defined-attribute-type-length INTEGER ::= 8
ub-domain-defined-attribute-value-length INTEGER ::= 128
ub-domain-name-length INTEGER ::= 16
ub-extension-attributes INTEGER ::= 256
ub-e163-4-number-length INTEGER ::= 15
ub-e163-4-sub-address-length INTEGER ::= 40
ub-generation-qualifier-length INTEGER ::= 3
ub-given-name-length INTEGER ::= 16
ub-initials-length INTEGER ::= 5
ub-integer-options INTEGER ::= 256
ub-numeric-user-id-length INTEGER ::= 32
ub-organization-name-length INTEGER ::= 64
ub-organizational-unit-name-length INTEGER ::= 32
ub-organizational-units INTEGER ::= 4
ub-pds-name-length INTEGER ::= 16
ub-pds-parameter-length INTEGER ::= 30
ub-pds-physical-address-lines INTEGER ::= 6
ub-postal-code-length INTEGER ::= 16
ub-surname-length INTEGER ::= 40
ub-terminal-id-length INTEGER ::= 24
ub-unformatted-address-length INTEGER ::= 180
ub-x121-address-length INTEGER ::= 16

-- Note - upper bounds on string types, such as TeletexString, are
-- measured in characters.  Excepting PrintableString or IA5String, a
-- significantly greater number of octets will be required to hold
```

```
-- such a value.  As a minimum, 16 octets, or twice the specified upper
-- bound, whichever is the larger, should be allowed for TeletexString.
-- For UTF8String or UniversalString at least four times the upper
-- bound should be allowed.

END
```

A.2 Implicitly Tagged Module, 1988 Syntax

```
PKIX1Implicit88 {iso(1) identified-organization(3) dod(6) internet(1)
  security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit-88(2)}

DEFINITIONS IMPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

IMPORTS
        id-pkix, id-pe, id-qt, id-kp, id-qt-unotice, id-qt-cps,
            id-ad, id-ad-ocsp, id-ad-caIssuers,
            -- delete following line if "new" types are supported --
            BMPString, UniversalString, UTF8String, -- end "new" types
                ORAddress, Name, RelativeDistinguishedName,
                CertificateSerialNumber,
                CertificateList, AlgorithmIdentifier, ub-name,
                Attribute, DirectoryString
                FROM PKIX1Explicit88 {iso(1) identified-organization(3)
                dod(6) internet(1) security(5) mechanisms(5) pkix(7)
                id-mod(0) id-pkix1-explicit(1)};


-- ISO arc for standard certificate and CRL extensions

id-ce OBJECT IDENTIFIER  ::=  {joint-iso-ccitt(2) ds(5) 29}

-- authority key identifier OID and syntax

id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 35 }

AuthorityKeyIdentifier ::= SEQUENCE {
      keyIdentifier             [0] KeyIdentifier            OPTIONAL,
      authorityCertIssuer       [1] GeneralNames             OPTIONAL,
      authorityCertSerialNumber [2] CertificateSerialNumber  OPTIONAL }
    -- authorityCertIssuer and authorityCertSerialNumber shall both
    -- be present or both be absent

KeyIdentifier ::= OCTET STRING

-- subject key identifier OID and syntax

id-ce-subjectKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 14 }

SubjectKeyIdentifier ::= KeyIdentifier
```

```
-- key usage extension OID and syntax

id-ce-keyUsage OBJECT IDENTIFIER ::=  { id-ce 15 }

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

-- private key usage period extension OID and syntax

id-ce-privateKeyUsagePeriod OBJECT IDENTIFIER ::=  { id-ce 16 }

PrivateKeyUsagePeriod ::= SEQUENCE {
     notBefore       [0]     GeneralizedTime OPTIONAL,
     notAfter        [1]     GeneralizedTime OPTIONAL }
     -- either notBefore or notAfter shall be present

-- certificate policies extension OID and syntax

id-ce-certificatePolicies OBJECT IDENTIFIER ::=  { id-ce 32 }

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

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

CertPolicyId ::= OBJECT IDENTIFIER

PolicyQualifierInfo ::= SEQUENCE {
       policyQualifierId  PolicyQualifierId,
       qualifier          ANY DEFINED BY policyQualifierId }

-- Implementations that recognize additional policy qualifiers shall
-- augment the following definition for PolicyQualifierId

PolicyQualifierId ::=
    OBJECT IDENTIFIER ( id-qt-cps | id-qt-unotice )

-- CPS pointer qualifier
```

```
CPSuri ::= IA5String

-- user notice qualifier

UserNotice ::= SEQUENCE {
     noticeRef        NoticeReference OPTIONAL,
     explicitText     DisplayText OPTIONAL}

NoticeReference ::= SEQUENCE {
     organization     DisplayText,
     noticeNumbers    SEQUENCE OF INTEGER }

DisplayText ::= CHOICE {
     visibleString    VisibleString  (SIZE (1..200)),
     bmpString        BMPString      (SIZE (1..200)),
     utf8String       UTF8String     (SIZE (1..200)) }

-- policy mapping extension OID and syntax

id-ce-policyMappings OBJECT IDENTIFIER ::=  { id-ce 33 }

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

-- subject alternative name extension OID and syntax

id-ce-subjectAltName OBJECT IDENTIFIER ::=  { id-ce 17 }

SubjectAltName ::= GeneralNames

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

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

-- AnotherName replaces OTHER-NAME ::= TYPE-IDENTIFIER, as
-- TYPE-IDENTIFIER is not supported in the '88 ASN.1 syntax

AnotherName ::= SEQUENCE {
```

```
     type-id    OBJECT IDENTIFIER,
     value      [0] EXPLICIT ANY DEFINED BY type-id }

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

-- issuer alternative name extension OID and syntax

id-ce-issuerAltName OBJECT IDENTIFIER ::=  { id-ce 18 }

IssuerAltName ::= GeneralNames

id-ce-subjectDirectoryAttributes OBJECT IDENTIFIER ::=  { id-ce 9 }

SubjectDirectoryAttributes ::= SEQUENCE SIZE (1..MAX) OF Attribute

-- basic constraints extension OID and syntax

id-ce-basicConstraints OBJECT IDENTIFIER ::=  { id-ce 19 }

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

-- name constraints extension OID and syntax

id-ce-nameConstraints OBJECT IDENTIFIER ::=  { id-ce 30 }

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

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

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

BaseDistance ::= INTEGER (0..MAX)

-- policy constraints extension OID and syntax

id-ce-policyConstraints OBJECT IDENTIFIER ::=  { id-ce 36 }

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

```
     inhibitPolicyMapping            [1] SkipCerts OPTIONAL }

SkipCerts ::= INTEGER (0..MAX)

-- CRL distribution points extension OID and syntax

id-ce-cRLDistributionPoints     OBJECT IDENTIFIER  ::=  {id-ce 31}

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) }

-- extended key usage extension OID and syntax

id-ce-extKeyUsage OBJECT IDENTIFIER ::= {id-ce 37}

ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId

KeyPurposeId ::= OBJECT IDENTIFIER

-- extended key purpose OIDs
id-kp-serverAuth      OBJECT IDENTIFIER ::= { id-kp 1 }
id-kp-clientAuth      OBJECT IDENTIFIER ::= { id-kp 2 }
id-kp-codeSigning     OBJECT IDENTIFIER ::= { id-kp 3 }
id-kp-emailProtection OBJECT IDENTIFIER ::= { id-kp 4 }
id-kp-ipsecEndSystem  OBJECT IDENTIFIER ::= { id-kp 5 }
id-kp-ipsecTunnel     OBJECT IDENTIFIER ::= { id-kp 6 }
id-kp-ipsecUser       OBJECT IDENTIFIER ::= { id-kp 7 }
id-kp-timeStamping    OBJECT IDENTIFIER ::= { id-kp 8 }

-- authority info access
```

```
id-pe-authorityInfoAccess OBJECT IDENTIFIER ::= { id-pe 1 }

AuthorityInfoAccessSyntax  ::=
        SEQUENCE SIZE (1..MAX) OF AccessDescription

AccessDescription  ::=  SEQUENCE {
        accessMethod          OBJECT IDENTIFIER,
        accessLocation        GeneralName  }

-- CRL number extension OID and syntax

id-ce-cRLNumber OBJECT IDENTIFIER ::= { id-ce 20 }

CRLNumber ::= INTEGER (0..MAX)

-- issuing distribution point extension OID and syntax

id-ce-issuingDistributionPoint OBJECT IDENTIFIER ::= { id-ce 28 }

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


id-ce-deltaCRLIndicator OBJECT IDENTIFIER ::= { id-ce 27 }

-- deltaCRLIndicator ::= BaseCRLNumber

BaseCRLNumber ::= CRLNumber

-- CRL reasons extension OID and syntax

id-ce-cRLReasons OBJECT IDENTIFIER ::= { id-ce 21 }

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

-- certificate issuer CRL entry extension OID and syntax
```

```
id-ce-certificateIssuer OBJECT IDENTIFIER ::= { id-ce 29 }

CertificateIssuer ::= GeneralNames

-- hold instruction extension OID and syntax

id-ce-holdInstructionCode OBJECT IDENTIFIER ::= { id-ce 23 }

HoldInstructionCode ::= OBJECT IDENTIFIER

-- ANSI x9 holdinstructions

-- ANSI x9 arc holdinstruction arc
holdInstruction OBJECT IDENTIFIER ::=
          {joint-iso-itu-t(2) member-body(2) us(840) x9cm(10040) 2}

-- ANSI X9 holdinstructions referenced by this standard
id-holdinstruction-none OBJECT IDENTIFIER  ::=
                {holdInstruction 1} -- deprecated
id-holdinstruction-callissuer OBJECT IDENTIFIER ::=
                {holdInstruction 2}
id-holdinstruction-reject OBJECT IDENTIFIER ::=
                {holdInstruction 3}

-- invalidity date CRL entry extension OID and syntax

id-ce-invalidityDate OBJECT IDENTIFIER ::= { id-ce 24 }

InvalidityDate ::=  GeneralizedTime

END
```

Appendix B. 1993 ASN.1 Structures and OIDs

B.1 Explicitly Tagged Module, 1993 Syntax

```
PKIX1Explicit93 {iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-explicit-93(3)}


DEFINITIONS EXPLICIT TAGS ::=

BEGIN

-- EXPORTS ALL --

IMPORTS
        authorityKeyIdentifier, subjectKeyIdentifier, keyUsage,
           extendedKeyUsage, privateKeyUsagePeriod, certificatePolicies,
           policyMappings, subjectAltName, issuerAltName,
           basicConstraints, nameConstraints, policyConstraints,
           cRLDistributionPoints, subjectDirectoryAttributes,
           cRLNumber, reasonCode, instructionCode, invalidityDate,
           issuingDistributionPoint, certificateIssuer,
           deltaCRLIndicator, authorityInfoAccess, id-ce
           FROM PKIX1Implicit93 {iso(1) identified-organization(3)
           dod(6) internet(1) security(5) mechanisms(5) pkix(7)
           id-mod(0) id-pkix1-implicit-93(4)} ;

--
                  --  Locally defined OIDs  --

id-pkix  OBJECT IDENTIFIER  ::=
         { iso(1) identified-organization(3) dod(6) internet(1)
                    security(5) mechanisms(5) pkix(7) }

-- PKIX arcs
-- arc for private certificate extensions
id-pe OBJECT IDENTIFIER  ::=  { id-pkix 1 }
 -- arc for policy qualifier types
id-qt OBJECT IDENTIFIER ::= { id-pkix 2 }
-- arc for extended key purpose OIDS
id-kp OBJECT IDENTIFIER ::= { id-pkix 3 }
-- arc for access descriptors
id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }

-- policyQualifierIds for Internet policy qualifiers
id-qt-cps      OBJECT IDENTIFIER ::=  { id-qt 1 }
        -- OID for CPS qualifier
```

```
id-qt-unotice  OBJECT IDENTIFIER ::=  { id-qt 2 }
        -- OID for user notice qualifier

-- based on excerpts from AuthenticationFramework
--    {joint-iso-ccitt ds(5) modules(1) authenticationFramework(7) 2}

               -- Public Key Certificate --

Certificate            ::=   SIGNED { SEQUENCE {
   version                 [0]   Version DEFAULT v1,
   serialNumber                  CertificateSerialNumber,
   signature                     AlgorithmIdentifier,
   issuer                        Name,
   validity                      Validity,
   subject                       Name,
   subjectPublicKeyInfo          SubjectPublicKeyInfo,
   issuerUniqueIdentifier  [1]   IMPLICIT UniqueIdentifier OPTIONAL,
                                    ---if present, version shall be v2 or v3--
   subjectUniqueIdentifier [2]   IMPLICIT UniqueIdentifier OPTIONAL,
                                   ---if present, version shall be v2 or v3--
   extensions              [3]   Extensions OPTIONAL
                                    --if present, version shall be v3--}  }

UniqueIdentifier        ::=  BIT STRING

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

CertificateSerialNumber ::=  INTEGER

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

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

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

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

Extension         ::=   SEQUENCE {
   extnId            EXTENSION.&id ({ExtensionSet}),
   critical          BOOLEAN DEFAULT FALSE,
   extnValue         OCTET STRING }
                -- contains a DER encoding of a value of type
```

```
                -- &ExtnType for the
                -- extension object identified by extnId --

-- The following information object set is defined to constrain the
-- set of legal certificate extensions.

ExtensionSet    EXTENSION       ::=     { authorityKeyIdentifier |
                                          subjectKeyIdentifier |
                                          keyUsage |
                                          extendedKeyUsage |
                                          privateKeyUsagePeriod |
                                          certificatePolicies |
                                          policyMappings |
                                          subjectAltName |
                                          issuerAltName |
                                          basicConstraints |
                                          nameConstraints |
                                          policyConstraints |
                                          cRLDistributionPoints |
                                          subjectDirectoryAttributes |
                                          authorityInfoAccess }

EXTENSION       ::=     CLASS {
   &id          OBJECT IDENTIFIER UNIQUE,
   &ExtnType }
WITH SYNTAX  {
   SYNTAX               &ExtnType
   IDENTIFIED BY        &id }

                  -- Certificate Revocation List --

CertificateList ::=    SIGNED { SEQUENCE {
   version                 Version  OPTIONAL, -- if present, shall be v2
   signature               AlgorithmIdentifier,
   issuer                  Name,
   thisUpdate              Time,
   nextUpdate              Time OPTIONAL,
   revokedCertificates     SEQUENCE OF SEQUENCE {
   userCertificate         CertificateSerialNumber,
   revocationDate          Time,
   crlEntryExtensions      EntryExtensions OPTIONAL } OPTIONAL,
   crlExtensions           [0]   CRLExtensions OPTIONAL }}

CRLExtensions        ::=        SEQUENCE SIZE (1..MAX) OF CRLExtension

CRLExtension         ::=        SEQUENCE {
   extnId            EXTENSION.&id ({CRLExtensionSet}),
   critical          BOOLEAN DEFAULT FALSE,
```

```
   extnValue         OCTET STRING }
                -- contains a DER encoding of a value of type
                -- &ExtnType for the
                -- extension object identified by extnId --

-- The following information object set is defined to constrain the
-- set of legal CRL extensions.

CRLExtensionSet EXTENSION       ::=     { authorityKeyIdentifier |
                                          issuerAltName |
                                          cRLNumber |
                                          deltaCRLIndicator |
                                          issuingDistributionPoint }

-- EXTENSION defined above for certificates

EntryExtensions        ::=      SEQUENCE SIZE (1..MAX) OF EntryExtension

EntryExtension         ::=      SEQUENCE {
   extnId            EXTENSION.&id ({EntryExtensionSet}),
   critical          BOOLEAN DEFAULT FALSE,
   extnValue         OCTET STRING }
                -- contains a DER encoding of a value of type
                -- &ExtnType for the
                -- extension object identified by extnId --

-- The following information object set is defined to constrain the
-- set of legal CRL entry extensions.

EntryExtensionSet       EXTENSION       ::=     { reasonCode |
                                                  instructionCode |
                                                  invalidityDate |
                                                  certificateIssuer }

         -- information object classes used in the defintion --
                    -- of certificates and CRLs --

-- Parameterized Type SIGNED --

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

-- Definition of AlgorithmIdentifier
-- ISO definition was:
--
```

```
-- AlgorithmIdentifier     ::=  SEQUENCE {
--   algorithm          ALGORITHM.&id({SupportedAlgorithms}),
--   parameters         ALGORITHM.&Type({SupportedAlgorithms}
--                                      { @algorithm}) OPTIONAL }
-- Definition of ALGORITHM
-- ALGORITHM    ::=     TYPE-IDENTIFIER

-- The following PKIX definition replaces the X.509 definition
--

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

-- Definition of ALGORITHM-ID

 ALGORITHM-ID ::= CLASS {
     &id    OBJECT IDENTIFIER UNIQUE,
     &Type  OPTIONAL
  }
     WITH SYNTAX { OID &id [PARMS &Type] }

-- The definition of SupportedAlgorithms may be modified as this
-- document does not specify a mandatory algorithm set.  In addition,
-- the set is specified as extensible, since additional algorithms
-- may be supported

SupportedAlgorithms     ALGORITHM-ID  ::=       { ..., -- extensible
                                          rsaPublicKey |
                                          rsaSHA-1  |
                                          rsaMD5 |
                                          rsaMD2 |
                                          dssPublicKey |
                                          dsaSHA-1 |
                                          dhPublicKey }

-- OIDs and parameter structures for ALGORITHM-IDs used
-- in this specification

rsaPublicKey ALGORITHM-ID ::= { OID rsaEncryption PARMS NULL }

rsaSHA-1 ALGORITHM-ID ::= { OID sha1WithRSAEncryption PARMS NULL }

rsaMD5 ALGORITHM-ID ::= { OID md5WithRSAEncryption PARMS NULL  }

rsaMD2 ALGORITHM-ID ::= { OID md2WithRSAEncryption PARMS NULL  }
```

```
dssPublicKey ALGORITHM-ID ::= { OID id-dsa PARMS Dss-Parms }

dsaSHA-1 ALGORITHM-ID ::= { OID id-dsa-with-sha1 }

dhPublicKey ALGORITHM-ID ::= {OID dhpublicnumber PARMS DomainParameters}

-- algorithm identifiers and parameter structures

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

rsaEncryption OBJECT IDENTIFIER ::=  { pkcs-1 1 }

md2WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 2 }

md5WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 4 }

sha1WithRSAEncryption OBJECT IDENTIFIER  ::=  { pkcs-1 5 }

id-dsa-with-sha1 OBJECT IDENTIFIER ::=  {
     iso(1) member-body(2) us(840) x9-57 (10040) x9algorithm(4) 3 }

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

dhpublicnumber OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) ansi-x942(10046) number-type(2) 1 }

DomainParameters ::= SEQUENCE {
     p       INTEGER, -- odd prime, p=jq +1
     g       INTEGER, -- generator, g
     q       INTEGER, -- factor of p-1
     j       INTEGER OPTIONAL, -- subgroup factor, j>= 2
     validationParms  ValidationParms OPTIONAL }

ValidationParms ::= SEQUENCE {
     seed             BIT STRING,
     pgenCounter      INTEGER }

id-dsa OBJECT IDENTIFIER ::= {
     iso(1) member-body(2) us(840) x9-57(10040) x9algorithm(4) 1 }

Dss-Parms  ::=  SEQUENCE  {
     p             INTEGER,
     q             INTEGER,
     g             INTEGER  }
```

```
     -- The ASN.1 in this section supports the Name type
     -- and the directoryAttribute extension

-- attribute data types --

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

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

-- naming data types --

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

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

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

ID     ::=    OBJECT IDENTIFIER

-- ATTRIBUTE information object class specification
--  Note: This has been greatly simplified for PKIX !!

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

-- suggested naming attributes
--      Definition of the following information object set may be
--    augmented to meet local requirements.  Note that deleting
--    members of the set may prevent interoperability with
--    conforming implementations.

SupportedAttributes     ATTRIBUTE       ::=     {
                name | commonName | surname | givenName | initials |
                generationQualifier | dnQualifier | countryName |
                localityName | stateOrProvinceName | organizationName |
                        organizationalUnitName | title | pkcs9email }

name ATTRIBUTE  ::=     {
```

```
        WITH SYNTAX                     DirectoryString { ub-name }
        ID                              id-at-name }

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

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

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

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

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

dnQualifier ATTRIBUTE   ::=     {
        WITH SYNTAX                     PrintableString
        ID                              id-at-dnQualifier }


countryName ATTRIBUTE   ::=     {
        WITH SYNTAX                     PrintableString (SIZE (2))
                                                -- IS 3166 codes only
        ID                              id-at-countryName }

localityName ATTRIBUTE  ::=     {
        WITH SYNTAX             DirectoryString {ub-locality-name}
        ID                      id-at-localityName }

stateOrProvinceName ATTRIBUTE   ::=     {
        WITH SYNTAX             DirectoryString {ub-state-name}
        ID                      id-at-stateOrProvinceName }

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 }
```

```
title ATTRIBUTE ::=                     {
        WITH SYNTAX             DirectoryString {ub-title}
        ID                      id-at-title }

 -- Legacy attributes

pkcs9email ATTRIBUTE ::= {
        WITH SYNTAX                     PHGString,
        ID                              emailAddress }

PHGString ::= IA5String (SIZE(1..ub-emailaddress-length))

pkcs-9 OBJECT IDENTIFIER ::=
       { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) 9 }

emailAddress OBJECT IDENTIFIER ::= { pkcs-9 1 }

    -- object identifiers for Name type and directory attribute support

-- Object identifier assignments --

id-at   OBJECT IDENTIFIER       ::=     {joint-iso-ccitt(2) ds(5) 4}

-- Attributes --

id-at-commonName        OBJECT IDENTIFIER       ::=     {id-at 3}
id-at-surname           OBJECT IDENTIFIER       ::=     {id-at 4}
id-at-countryName       OBJECT IDENTIFIER       ::=     {id-at 6}
id-at-localityName      OBJECT IDENTIFIER       ::=     {id-at 7}
id-at-stateOrProvinceName     OBJECT IDENTIFIER ::= {id-at 8}
id-at-organizationName        OBJECT IDENTIFIER ::= {id-at 10}
id-at-organizationalUnitName  OBJECT IDENTIFIER ::= {id-at 11}
id-at-title             OBJECT IDENTIFIER       ::=     {id-at 12}
id-at-name              OBJECT IDENTIFIER       ::=     {id-at 41}
id-at-givenName         OBJECT IDENTIFIER       ::=     {id-at 42}
id-at-initials          OBJECT IDENTIFIER       ::=     {id-at 43}
id-at-generationQualifier   OBJECT IDENTIFIER   ::=     {id-at 44}
id-at-dnQualifier       OBJECT IDENTIFIER       ::=     {id-at 46}

-- Directory string type, used extensively in Name types --

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))
                        }
```

```
     -- End of ASN.1 for Name type and directory attribute support --

     -- The ASN.1 in this section supports X.400 style names   --
     -- for implementations that use the x400Address component --
     -- of GeneralName.                                        --

ORAddress ::= SEQUENCE {
   built-in-standard-attributes BuiltInStandardAttributes,
   built-in-domain-defined-attributes
                        BuiltInDomainDefinedAttributes OPTIONAL,
   -- see also teletex-domain-defined-attributes
   extension-attributes ExtensionAttributes OPTIONAL }

--  The OR-address is semantically absent from the OR-name if the
--  built-in-standard-attribute sequence is empty and the
--  built-in-domain-defined-attributes and extension-attributes are
--  both omitted.

--      Built-in Standard Attributes

BuiltInStandardAttributes ::= SEQUENCE {
   country-name CountryName OPTIONAL,
   administration-domain-name AdministrationDomainName OPTIONAL,
   network-address      [0] NetworkAddress OPTIONAL,
   -- see also extended-network-address
   terminal-identifier  [1] TerminalIdentifier OPTIONAL,
   private-domain-name  [2] PrivateDomainName OPTIONAL,
   organization-name    [3] OrganizationName OPTIONAL,
   -- see also teletex-organization-name
   numeric-user-identifier      [4] NumericUserIdentifier OPTIONAL,
   personal-name        [5] PersonalName OPTIONAL,
   -- see also teletex-personal-name
   organizational-unit-names    [6] OrganizationalUnitNames OPTIONAL
   -- see also teletex-organizational-unit-names -- }

CountryName ::= [APPLICATION 1] CHOICE {
   x121-dcc-code NumericString
               (SIZE (ub-country-name-numeric-length)),
   iso-3166-alpha2-code PrintableString
               (SIZE (ub-country-name-alpha-length)) }

AdministrationDomainName ::= [APPLICATION 2] CHOICE {
   numeric NumericString (SIZE (0..ub-domain-name-length)),
   printable PrintableString (SIZE (0..ub-domain-name-length)) }

NetworkAddress ::= X121Address
-- see also extended-network-address
```

```
X121Address ::= NumericString (SIZE (1..ub-x121-address-length))

TerminalIdentifier ::= PrintableString (SIZE (1..ub-terminal-id-length))

PrivateDomainName ::= CHOICE {
   numeric NumericString (SIZE (1..ub-domain-name-length)),
   printable PrintableString (SIZE (1..ub-domain-name-length)) }

OrganizationName ::= PrintableString
                           (SIZE (1..ub-organization-name-length))
-- see also teletex-organization-name

NumericUserIdentifier ::= NumericString
                            (SIZE (1..ub-numeric-user-id-length))

PersonalName ::= SET {
   surname    [0] PrintableString (SIZE (1..ub-surname-length)),
   given-name [1] PrintableString
                       (SIZE (1..ub-given-name-length)) OPTIONAL,
   initials   [2] PrintableString
                       (SIZE (1..ub-initials-length)) OPTIONAL,
   generation-qualifier [3] PrintableString
                (SIZE (1..ub-generation-qualifier-length)) OPTIONAL}
-- see also teletex-personal-name

OrganizationalUnitNames ::= SEQUENCE SIZE (1..ub-organizational-units)
                                       OF OrganizationalUnitName
-- see also teletex-organizational-unit-names

OrganizationalUnitName ::= PrintableString (SIZE
                       (1..ub-organizational-unit-name-length))

--      Built-in Domain-defined Attributes
BuiltInDomainDefinedAttributes ::= SEQUENCE SIZE
                                (1..ub-domain-defined-attributes) OF
                                BuiltInDomainDefinedAttribute

BuiltInDomainDefinedAttribute ::= SEQUENCE {
   type PrintableString (SIZE
                (1..ub-domain-defined-attribute-type-length)),
   value PrintableString (SIZE
                (1..ub-domain-defined-attribute-value-length)) }

--      Extension Attributes

ExtensionAttributes ::= SET SIZE (1..ub-extension-attributes)
                                       OF ExtensionAttribute
ExtensionAttribute ::= SEQUENCE {
```

```
        extension-attribute-type [0] EXTENSION-ATTRIBUTE.&id
                                        ({ExtensionAttributeTable}),
        extension-attribute-value [1] EXTENSION-ATTRIBUTE.&Type
             ({ExtensionAttributeTable} {@extension-attribute-type}) }

EXTENSION-ATTRIBUTE ::= CLASS {
        &id     INTEGER (0..ub-extension-attributes) UNIQUE,
        &Type }
WITH SYNTAX {&Type IDENTIFIED BY &id}

ExtensionAttributeTable EXTENSION-ATTRIBUTE ::= {
        common-name |
        teletex-common-name |
        teletex-organization-name |
        teletex-personal-name |
        teletex-organizational-unit-names |
        teletex-domain-defined-attributes |
        pds-name |
        physical-delivery-country-name |
        postal-code |
        physical-delivery-office-name |
        physical-delivery-office-number |
        extension-OR-address-components |
        physical-delivery-personal-name |
        physical-delivery-organization-name |
        extension-physical-delivery-address-components |
        unformatted-postal-address |
        street-address |
        post-office-box-address |
        poste-restante-address |
        unique-postal-name |
        local-postal-attributes |
        extended-network-address |
        terminal-type }

--      Extension Standard Attributes

common-name EXTENSION-ATTRIBUTE ::= {CommonName IDENTIFIED BY 1}

CommonName ::= PrintableString (SIZE (1..ub-common-name-length))

teletex-common-name EXTENSION-ATTRIBUTE ::=
                {TeletexCommonName IDENTIFIED BY 2}

TeletexCommonName ::= TeletexString (SIZE (1..ub-common-name-length))

teletex-organization-name EXTENSION-ATTRIBUTE ::=
                {TeletexOrganizationName IDENTIFIED BY 3}
```

```
TeletexOrganizationName ::=
               TeletexString (SIZE (1..ub-organization-name-length))

teletex-personal-name EXTENSION-ATTRIBUTE ::=
               {TeletexPersonalName IDENTIFIED BY 4}

TeletexPersonalName ::= SET {
   surname [0] TeletexString (SIZE (1..ub-surname-length)),
   given-name [1] TeletexString
               (SIZE (1..ub-given-name-length)) OPTIONAL,
   initials [2] TeletexString (SIZE (1..ub-initials-length)) OPTIONAL,
   generation-qualifier [3] TeletexString (SIZE
               (1..ub-generation-qualifier-length)) OPTIONAL }

teletex-organizational-unit-names EXTENSION-ATTRIBUTE ::=
   {TeletexOrganizationalUnitNames IDENTIFIED BY 5}

TeletexOrganizationalUnitNames ::= SEQUENCE SIZE
        (1..ub-organizational-units) OF TeletexOrganizationalUnitName

TeletexOrganizationalUnitName ::= TeletexString
                        (SIZE (1..ub-organizational-unit-name-length))

pds-name EXTENSION-ATTRIBUTE ::= {PDSName IDENTIFIED BY 7}

PDSName ::= PrintableString (SIZE (1..ub-pds-name-length))

physical-delivery-country-name EXTENSION-ATTRIBUTE ::=
   {PhysicalDeliveryCountryName IDENTIFIED BY 8}

PhysicalDeliveryCountryName ::= CHOICE {
   x121-dcc-code NumericString (SIZE (ub-country-name-numeric-length)),
   iso-3166-alpha2-code PrintableString
                        (SIZE (ub-country-name-alpha-length)) }

postal-code EXTENSION-ATTRIBUTE ::= {PostalCode IDENTIFIED BY 9}

PostalCode ::= CHOICE {
   numeric-code NumericString (SIZE (1..ub-postal-code-length)),
   printable-code PrintableString (SIZE (1..ub-postal-code-length)) }

physical-delivery-office-name EXTENSION-ATTRIBUTE ::=
                        {PhysicalDeliveryOfficeName IDENTIFIED BY 10}

PhysicalDeliveryOfficeName ::= PDSParameter

physical-delivery-office-number EXTENSION-ATTRIBUTE ::=
   {PhysicalDeliveryOfficeNumber IDENTIFIED BY 11}
```

```
PhysicalDeliveryOfficeNumber ::= PDSParameter

extension-OR-address-components EXTENSION-ATTRIBUTE ::=
   {ExtensionORAddressComponents IDENTIFIED BY 12}

ExtensionORAddressComponents ::= PDSParameter

physical-delivery-personal-name EXTENSION-ATTRIBUTE ::=
   {PhysicalDeliveryPersonalName IDENTIFIED BY 13}

PhysicalDeliveryPersonalName ::= PDSParameter

physical-delivery-organization-name EXTENSION-ATTRIBUTE ::=
   {PhysicalDeliveryOrganizationName IDENTIFIED BY 14}

PhysicalDeliveryOrganizationName ::= PDSParameter

extension-physical-delivery-address-components EXTENSION-ATTRIBUTE ::=
   {ExtensionPhysicalDeliveryAddressComponents IDENTIFIED BY 15}

ExtensionPhysicalDeliveryAddressComponents ::= PDSParameter

unformatted-postal-address EXTENSION-ATTRIBUTE ::=
                        {UnformattedPostalAddress IDENTIFIED BY 16}

UnformattedPostalAddress ::= SET {
   printable-address SEQUENCE SIZE (1..ub-pds-physical-address-lines) OF
           PrintableString (SIZE (1..ub-pds-parameter-length)) OPTIONAL,
   teletex-string TeletexString (SIZE
                         (1..ub-unformatted-address-length)) OPTIONAL }

street-address EXTENSION-ATTRIBUTE ::=
                {StreetAddress IDENTIFIED BY 17}

StreetAddress ::= PDSParameter

post-office-box-address EXTENSION-ATTRIBUTE ::=
                {PostOfficeBoxAddress IDENTIFIED BY 18}

PostOfficeBoxAddress ::= PDSParameter

poste-restante-address EXTENSION-ATTRIBUTE ::=
                {PosteRestanteAddress IDENTIFIED BY 19}

PosteRestanteAddress ::= PDSParameter

unique-postal-name EXTENSION-ATTRIBUTE ::=
                {UniquePostalName IDENTIFIED BY 20}
```

```
UniquePostalName ::= PDSParameter

local-postal-attributes EXTENSION-ATTRIBUTE ::=
                {LocalPostalAttributes IDENTIFIED BY 21}

LocalPostalAttributes ::= PDSParameter

PDSParameter ::= SET {
   printable-string PrintableString
            (SIZE(1..ub-pds-parameter-length)) OPTIONAL,
   teletex-string TeletexString
            (SIZE(1..ub-pds-parameter-length)) OPTIONAL }

extended-network-address EXTENSION-ATTRIBUTE ::=
                {ExtendedNetworkAddress IDENTIFIED BY 22}

ExtendedNetworkAddress ::= CHOICE {
        e163-4-address SEQUENCE {
                number [0] NumericString
                   (SIZE (1..ub-e163-4-number-length)),
                sub-address [1] NumericString
                   (SIZE (1..ub-e163-4-sub-address-length)) OPTIONAL},
        psap-address [0] PresentationAddress }

PresentationAddress ::= SEQUENCE {
        pSelector       [0] EXPLICIT OCTET STRING OPTIONAL,
        sSelector       [1] EXPLICIT OCTET STRING OPTIONAL,
        tSelector       [2] EXPLICIT OCTET STRING OPTIONAL,
        nAddresses      [3] EXPLICIT SET SIZE (1..MAX) OF OCTET STRING}


terminal-type EXTENSION-ATTRIBUTE ::= {TerminalType IDENTIFIED BY 23}

TerminalType ::= INTEGER {
   telex (3),
   teletex (4),
   g3-facsimile (5),
   g4-facsimile (6),
   ia5-terminal (7),
   videotex (8) } (0..ub-integer-options)

--      Extension Domain-defined Attributes

teletex-domain-defined-attributes EXTENSION-ATTRIBUTE ::=
   {TeletexDomainDefinedAttributes IDENTIFIED BY 6}

TeletexDomainDefinedAttributes ::= SEQUENCE SIZE
   (1..ub-domain-defined-attributes) OF TeletexDomainDefinedAttribute
```

```
TeletexDomainDefinedAttribute ::= SEQUENCE {
    type TeletexString
         (SIZE (1..ub-domain-defined-attribute-type-length)),
    value TeletexString
         (SIZE (1..ub-domain-defined-attribute-value-length)) }

--  specifications of Upper Bounds
--  shall be regarded as mandatory
--  from Annex B of ITU-T X.411
--  Reference Definition of MTS Parameter Upper Bounds

--      Upper Bounds
ub-name INTEGER ::=     32768
ub-common-name  INTEGER ::=     64
ub-locality-name        INTEGER ::=     128
ub-state-name   INTEGER ::=     128
ub-organization-name    INTEGER ::=     64
ub-organizational-unit-name     INTEGER ::=     64
ub-title        INTEGER ::=     64
ub-match        INTEGER ::=     128

ub-emailaddress-length INTEGER ::= 128

ub-common-name-length INTEGER ::= 64
ub-country-name-alpha-length INTEGER ::= 2
ub-country-name-numeric-length INTEGER ::= 3
ub-domain-defined-attributes INTEGER ::= 4
ub-domain-defined-attribute-type-length INTEGER ::= 8
ub-domain-defined-attribute-value-length INTEGER ::= 128
ub-domain-name-length INTEGER ::= 16
ub-extension-attributes INTEGER ::= 256
ub-e163-4-number-length INTEGER ::= 15
ub-e163-4-sub-address-length INTEGER ::= 40
ub-generation-qualifier-length INTEGER ::= 3
ub-given-name-length INTEGER ::= 16
ub-initials-length INTEGER ::= 5
ub-integer-options INTEGER ::= 256
ub-numeric-user-id-length INTEGER ::= 32
ub-organization-name-length INTEGER ::= 64
ub-organizational-unit-name-length INTEGER ::= 32
ub-organizational-units INTEGER ::= 4
ub-pds-name-length INTEGER ::= 16
ub-pds-parameter-length INTEGER ::= 30
ub-pds-physical-address-lines INTEGER ::= 6
ub-postal-code-length INTEGER ::= 16
ub-surname-length INTEGER ::= 40
ub-terminal-id-length INTEGER ::= 24
ub-unformatted-address-length INTEGER ::= 180
```

```
ub-x121-address-length INTEGER ::= 16

-- Note - upper bounds on TeletexString are measured in characters.
-- A significantly greater number of octets will be required to hold
-- such a value.  As a minimum, 16 octets, or twice the specified upper
-- bound, whichever is the larger, should be allowed.

END
```

B.2 Implicitly Tagged Module, 1993 Syntax

```
PKIX1Implicit93  {iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit-93(4)}

DEFINITIONS IMPLICIT TAGS::=

BEGIN

--EXPORTS ALL --

IMPORTS
        id-pe, id-qt, id-kp, id-ad, id-qt-unotice,
                ORAddress, Name, RelativeDistinguishedName,
                CertificateSerialNumber, CertificateList,
                AlgorithmIdentifier, ub-name, DirectoryString,
                Attribute, EXTENSION
                FROM PKIX1Explicit93 {iso(1) identified-organization(3)
                dod(6) internet(1) security(5) mechanisms(5) pkix(7)
                id-mod(0) id-pkix1-explicit-93(3)};

-- Key and policy information extensions --

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

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

KeyIdentifier ::= OCTET STRING

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

SubjectKeyIdentifier ::= KeyIdentifier

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

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

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

KeyPurposeId ::= OBJECT IDENTIFIER

-- PKIX-defined extended key purpose OIDs
id-kp-serverAuth      OBJECT IDENTIFIER ::= { id-kp 1 }
id-kp-clientAuth      OBJECT IDENTIFIER ::= { id-kp 2 }
id-kp-codeSigning     OBJECT IDENTIFIER ::= { id-kp 3 }
id-kp-emailProtection OBJECT IDENTIFIER ::= { id-kp 4 }
id-kp-ipsecEndSystem  OBJECT IDENTIFIER ::= { id-kp 5 }
id-kp-ipsecTunnel     OBJECT IDENTIFIER ::= { id-kp 6 }
id-kp-ipsecUser       OBJECT IDENTIFIER ::= { id-kp 7 }
id-kp-timeStamping    OBJECT IDENTIFIER ::= { id-kp 8 }

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
        IDENTIFIED BY id-ce-certificatePolicies }

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

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

```
CertPolicyId ::= OBJECT IDENTIFIER

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

SupportedPolicyQualifiers CERT-POLICY-QUALIFIER ::= { noticeToUser |
                                                      pointerToCPS }

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

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

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

-- Certificate subject and certificate issuer attributes extensions --

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

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

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

OTHER-NAME ::= TYPE-IDENTIFIER
```

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

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

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

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

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

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

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

BaseDistance ::= INTEGER (0..MAX)

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

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

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

-- 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) }

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

HoldInstruction ::= OBJECT IDENTIFIER

-- holdinstructions described in this specification, from ANSI x9

-- ANSI x9 arc holdinstruction arc
holdInstruction OBJECT IDENTIFIER ::= {
     joint-iso-ccitt(2) member-body(2) us(840) x9cm(10040) 2}

-- ANSI X9 holdinstructions referenced by this standard
id-holdinstruction-none OBJECT IDENTIFIER ::= {holdInstruction 1}
id-holdinstruction-callissuer OBJECT IDENTIFIER ::= {holdInstruction 2}
id-holdinstruction-reject OBJECT IDENTIFIER ::= {holdInstruction 3}

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

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

cRLDistributionPoints EXTENSION ::= {
```

```
        SYNTAX  CRLDistPointsSyntax
        IDENTIFIED BY id-ce-cRLDistributionPoints }

CRLDistPointsSyntax ::= SEQUENCE SIZE (1..MAX) OF DistributionPoint

DistributionPoint ::= SEQUENCE {
        distributionPoint       [0]     DistributionPointName OPTIONAL,
        reasons         [1]     ReasonFlags OPTIONAL,
        cRLIssuer               [2]     GeneralNames OPTIONAL }

DistributionPointName ::= CHOICE {
        fullName                [0]     GeneralNames,
        nameRelativeToCRLIssuer [1]     RelativeDistinguishedName }

ReasonFlags ::= BIT STRING {
        unused                  (0),
        keyCompromise           (1),
        caCompromise            (2),
        affiliationChanged      (3),
        superseded              (4),
        cessationOfOperation    (5),
        certificateHold         (6) }

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

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

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

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

BaseCRLNumber ::= CRLNumber

-- Object identifier assignments for ISO certificate extensions --
id-ce   OBJECT IDENTIFIER       ::=     {joint-iso-ccitt(2) ds(5) 29}

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}
id-ce-policyConstraints            OBJECT IDENTIFIER ::= {id-ce 36}
id-ce-authorityKeyIdentifier       OBJECT IDENTIFIER ::= {id-ce 35}
id-ce-extKeyUsage                  OBJECT IDENTIFIER ::= {id-ce 37}

-- PKIX 1 extensions

authorityInfoAccess EXTENSION ::= {
        SYNTAX  AuthorityInfoAccessSyntax
        IDENTIFIED BY id-pe-authorityInfoAccess }

AuthorityInfoAccessSyntax  ::=
        SEQUENCE SIZE (1..MAX) OF AccessDescription

AccessDescription  ::=  SEQUENCE {
        accessMethod          OBJECT IDENTIFIER,
        accessLocation        GeneralName  }

id-pe-authorityInfoAccess OBJECT IDENTIFIER ::= { id-pe 1 }

id-ad-ocsp      OBJECT IDENTIFIER ::= { id-ad 1 }
id-ad-caIssuers OBJECT IDENTIFIER ::= { id-ad 2 }

-- PKIX policy qualifier definitions

noticeToUser CERT-POLICY-QUALIFIER ::= {
     POLICY-QUALIFIER-ID    id-qt-cps QUALIFIER-TYPE       CPSuri}

pointerToCPS CERT-POLICY-QUALIFIER ::= {
     POLICY-QUALIFIER-ID    id-qt-unotice QUALIFIER-TYPE   UserNotice}

id-qt-cps      OBJECT IDENTIFIER ::=  { id-qt 1 }
```

```
id-qt-unotice  OBJECT IDENTIFIER ::=  { id-qt 2 }

CPSuri ::= IA5String

UserNotice ::= SEQUENCE {
     noticeRef        NoticeReference OPTIONAL,
     explicitText     DisplayText OPTIONAL}

NoticeReference ::= SEQUENCE {
     organization     DisplayText,
     noticeNumbers    SEQUENCE OF INTEGER }

DisplayText ::= CHOICE {
     visibleString    VisibleString  (SIZE (1..200)),
     bmpString        BMPString      (SIZE (1..200)),
     utf8String       UTF8String     (SIZE (1..200)) }


END
```

Appendix C. ASN.1 Notes

The construct "SEQUENCE SIZE (1..MAX) OF" appears in several ASN.1 constructs. A valid ASN.1 sequence will have zero or more entries. The SIZE (1..MAX) construct constrains the sequence to have at least one entry. MAX indicates the upper bound is unspecified. Implementations are free to choose an upper bound that suits their environment.

The construct "positiveInt ::= INTEGER (0..MAX)" defines positiveInt as a subtype of INTEGER containing integers greater than or equal to zero. The upper bound is unspecified. Implementations are free to select an upper bound that suits their environment.

The character string type PrintableString supports a very basic Latin character set: the lower case letters 'a' through 'z', upper case letters 'A' through 'Z', the digits '0' through '9', eleven special characters ' " ( ) + , - . / : ? and space.

The character string type TeletexString is a superset of PrintableString. TeletexString supports a fairly standard (ascii-like) Latin character set, Latin characters with non-spacing accents and Japanese characters.

The character string type UniversalString supports any of the characters allowed by ISO 10646-1. ISO 10646 is the Universal multiple-octet coded Character Set (UCS). ISO 10646-1 specifes the architecture and the "basic multilingual plane" - a large standard character set which includes all major world character standards.

The character string type UTF8String will be introduced in the 1998 version of ASN.1. UTF8String is a universal type and has been assigned tag number 12. The content of UTF8String was defined by RFC 2044 and updated in RFC 2279, "UTF-8, a transformation Format of ISP 10646." ISO is expected to formally add UTF8String to the list of choices for DirectoryString in 1998 as well.

In anticipation of these changes, and in conformance with IETF Best Practices codified in RFC 2277, IETF Policy on Character Sets and Languages, this document includes UTF8String as a choice in DirectoryString and the CPS qualifier extensions.

Appendix D. Examples

This section contains four examples: three certificates and a CRL. The first two certificates and the CRL comprise a minimal certification path.

Section D.1 contains an annotated hex dump of a "self-signed" certificate issued by a CA whose distinguished name is cn=us,o=gov,ou=nist. The certificate contains a DSA public key with parameters, and is signed by the corresponding DSA private key.

Section D.2 contains an annotated hex dump of an end-entity certificate. The end entity certificate contains a DSA public key, and is signed by the private key corresponding to the "self-signed" certificate in section D.1.

Section D.3 contains a dump of an end entity certificate which contains an RSA public key and is signed with RSA and MD5. This certificate is not part of the minimal certification path.

Section D.4 contains an annotated hex dump of a CRL. The CRL is issued by the CA whose distinguished name is cn=us,o=gov,ou=nist and the list of revoked certificates includes the end entity certificate presented in D.2.

D.1 Certificate

This section contains an annotated hex dump of a 699 byte version 3 certificate. The certificate contains the following information:
(a) the serial number is 17 (11 hex);
(b) the certificate is signed with DSA and the SHA-1 hash algorithm;
(c) the issuer’s distinguished name is OU=nist; O=gov; C=US
(d) and the subject’s distinguished name is OU=nist; O=gov; C=US
(e) the certificate was issued on June 30, 1997 and will expire on December 31, 1997;
(f) the certificate contains a 1024 bit DSA public key with parameters;
(g) the certificate contains a subject key identifier extension; and
(h) the certificate is a CA certificate (as indicated through the basic constraints extension.)

```
0000 30 82 02 b7  695: SEQUENCE
0004 30 82 02 77  631: . SEQUENCE    tbscertificate
0008 a0 03          3: . . [0]
0010 02 01          1: . . . INTEGER 2
                     : 02
0013 02 01          1: . . INTEGER 17
                     : 11
```

```
0016 30 09          9: . . SEQUENCE
0018 06 07          7: . . . OID 1.2.840.10040.4.3: dsa-with-sha
                     : 2a 86 48 ce 38 04 03
0027 30 2a         42: . . SEQUENCE
0029 31 0b         11: . . . SET
0031 30 09          9: . . . . SEQUENCE
0033 06 03          3: . . . . . OID 2.5.4.6: C
                     : 55 04 06
0038 13 02          2: . . . . . PrintableString  'US'
                     : 55 53
0042 31 0c         12: . . . SET
0044 30 0a         10: . . . . SEQUENCE
0046 06 03          3: . . . . . OID 2.5.4.10: O
                     : 55 04 0a
0051 13 03          3: . . . . . PrintableString  'gov'
                     : 67 6f 76
0056 31 0d         13: . . . SET
0058 30 0b         11: . . . . SEQUENCE
0060 06 03          3: . . . . . OID 2.5.4.11: OU
                     : 55 04 0b
0065 13 04          4: . . . . . PrintableString  'nist'
                     : 6e 69 73 74
0071 30 1e         30: . . SEQUENCE
0073 17 0d         13: . . . UTCTime  '970630000000Z'
                     : 39 37 30 36 33 30 30 30 30 30 30 30 5a
0088 17 0d         13: . . . UTCTime  '971231000000Z'
                     : 39 37 31 32 33 31 30 30 30 30 30 30 5a
0103 30 2a         42: . . SEQUENCE
0105 31 0b         11: . . . SET
0107 30 09          9: . . . . SEQUENCE
0109 06 03          3: . . . . . OID 2.5.4.6: C
                     : 55 04 06
0114 13 02          2: . . . . . PrintableString  'US'
                     : 55 53
0118 31 0c         12: . . . SET
0120 30 0a         10: . . . . SEQUENCE
0122 06 03          3: . . . . . OID 2.5.4.10: O
                     : 55 04 0a
0127 13 03          3: . . . . . PrintableString  'gov'
                     : 67 6f 76
0132 31 0d         13: . . . SET
0134 30 0b         11: . . . . SEQUENCE
0136 06 03          3: . . . . . OID 2.5.4.11: OU
                     : 55 04 0b
0141 13 04          4: . . . . . PrintableString  'nist'
                     : 6e 69 73 74
0147 30 82 01 b4  436: . . SEQUENCE
0151 30 82 01 29  297: . . . SEQUENCE
```

```
0155 06 07          7: . . . . OID 1.2.840.10040.4.1: dsa
                     : 2a 86 48 ce 38 04 01
0164 30 82 01 1c  284: . . . . SEQUENCE
0168 02 81 80     128: . . . . . INTEGER
                     : d4 38 02 c5 35 7b d5 0b a1 7e 5d 72 59 63 55 d3
                     : 45 56 ea e2 25 1a 6b c5 a4 ab aa 0b d4 62 b4 d2
                     : 21 b1 95 a2 c6 01 c9 c3 fa 01 6f 79 86 83 3d 03
                     : 61 e1 f1 92 ac bc 03 4e 89 a3 c9 53 4a f7 e2 a6
                     : 48 cf 42 1e 21 b1 5c 2b 3a 7f ba be 6b 5a f7 0a
                     : 26 d8 8e 1b eb ec bf 1e 5a 3f 45 c0 bd 31 23 be
                     : 69 71 a7 c2 90 fe a5 d6 80 b5 24 dc 44 9c eb 4d
                     : f9 da f0 c8 e8 a2 4c 99 07 5c 8e 35 2b 7d 57 8d
0299 02 14         20: . . . . . INTEGER
                     : a7 83 9b f3 bd 2c 20 07 fc 4c e7 e8 9f f3 39 83
                     : 51 0d dc dd
0321 02 81 80     128: . . . . . INTEGER
                     : 0e 3b 46 31 8a 0a 58 86 40 84 e3 a1 22 0d 88 ca
                     : 90 88 57 64 9f 01 21 e0 15 05 94 24 82 e2 10 90
                     : d9 e1 4e 10 5c e7 54 6b d4 0c 2b 1b 59 0a a0 b5
                     : a1 7d b5 07 e3 65 7c ea 90 d8 8e 30 42 e4 85 bb
                     : ac fa 4e 76 4b 78 0e df 6c e5 a6 e1 bd 59 77 7d
                     : a6 97 59 c5 29 a7 b3 3f 95 3e 9d f1 59 2d f7 42
                     : 87 62 3f f1 b8 6f c7 3d 4b b8 8d 74 c4 ca 44 90
                     : cf 67 db de 14 60 97 4a d1 f7 6d 9e 09 94 c4 0d
0452 03 81 84     132: . . . BIT STRING  (0 unused bits)
                     : 02 81 80 aa 98 ea 13 94 a2 db f1 5b 7f 98 2f 78
                     : e7 d8 e3 b9 71 86 f6 80 2f 40 39 c3 da 3b 4b 13
                     : 46 26 ee 0d 56 c5 a3 3a 39 b7 7d 33 c2 6b 5c 77
                     : 92 f2 55 65 90 39 cd 1a 3c 86 e1 32 eb 25 bc 91
                     : c4 ff 80 4f 36 61 bd cc e2 61 04 e0 7e 60 13 ca
                     : c0 9c dd e0 ea 41 de 33 c1 f1 44 a9 bc 71 de cf
                     : 59 d4 6e da 44 99 3c 21 64 e4 78 54 9d d0 7b ba
                     : 4e f5 18 4d 5e 39 30 bf e0 d1 f6 f4 83 25 4f 14
                     : aa 71 e1
0587 a3 32         50: . . [3]
0589 30 30         48: . . . SEQUENCE
0591 30 0f          9: . . . . SEQUENCE
0593 06 03          3: . . . . . OID 2.5.29.19: basicConstraints
                     : 55 1d 13
0598 01 01          1: . . . . . TRUE
                     : ff
0601 04 05          5: . . . . . OCTET STRING
                     : 30 03 01 01 ff
0608 30 1d         29: . SEQUENCE
0610 06 03          3: . . . . . OID 2.5.29.14: subjectKeyIdentifier
                     : 55 1d 0e
0615 04 16         22: . . . . . OCTET STRING
                     : 04 14 e7 26 c5 54 cd 5b a3 6f 35 68 95 aa d5 ff
```

```
                     : 1c 21 e4 22 75 d6
0639 30 09          9: . SEQUENCE
0641 06 07          7: . . OID 1.2.840.10040.4.3: dsa-with-sha
                     : 2a 86 48 ce 38 04 03
0650 03 2f         47: . BIT STRING  (0 unused bits)
                     : 30 2c 02 14 a0 66 c1 76 33 99 13 51 8d 93 64 2f
                     : ca 13 73 de 79 1a 7d 33 02 14 5d 90 f6 ce 92 4a
                     : bf 29 11 24 80 28 a6 5a 8e 73 b6 76 02 68
```

D.2 Certificate

This section contains an annotated hex dump of a 730 byte version 3 certificate. The certificate contains the following information: (a) the serial number is 18 (12 hex); (b) the certificate is signed with DSA and the SHA-1 hash algorithm; (c) the issuer's distinguished name is OU=nist; O=gov; C=US (d) and the subject's distinguished name is CN=Tim Polk; OU=nist; O=gov; C=US (e) the certificate was valid from July 30, 1997 through December 1, 1997; (f) the certificate contains a 1024 bit DSA public key; (g) the certificate is an end entity certificate, as the basic constraints extension is not present; (h) the certificate contains an authority key identifier extension; and (i) the certificate includes one alternative name - an RFC 822 address.

```
0000 30 82 02 d6  726: SEQUENCE
0004 30 82 02 96  662: . SEQUENCE
0008 a0 03          3: . . [0]
0010 02 01          1: . . . INTEGER 2
                     : 02
0013 02 01          1: . . INTEGER 18
                     : 12
0016 30 09          9: . . SEQUENCE
0018 06 07          7: . . . OID 1.2.840.10040.4.3: dsa-with-sha
                     : 2a 86 48 ce 38 04 03
0027 30 2a         42: . . SEQUENCE
0029 31 0b         11: . . . SET
0031 30 09          9: . . . . SEQUENCE
0033 06 03          3: . . . . . OID 2.5.4.6: C
                     : 55 04 06
0038 13 02          2: . . . . . PrintableString  'US'
                     : 55 53
0042 31 0c         12: . . . SET
0044 30 0a         10: . . . . SEQUENCE
0046 06 03          3: . . . . . OID 2.5.4.10: O
```

```
                      : 55 04 0a
0051 13 03           3: . . . . . PrintableString  'gov'
                      : 67 6f 76
0056 31 0d          13: . . . SET
0058 30 0b          11: . . . . SEQUENCE
0060 06 03           3: . . . . . OID 2.5.4.11: OU
                      : 55 04 0b
0065 13 04           4: . . . . . PrintableString  'nist'
                      : 6e 69 73 74
0071 30 1e          30: . . SEQUENCE
0073 17 0d          13: . . . UTCTime  '970730000000Z'
                      : 39 37 30 37 33 30 30 30 30 30 30 30 5a
0088 17 0d          13: . . . UTCTime  '971201000000Z'
                      : 39 37 31 32 30 31 30 30 30 30 30 30 5a
0103 30 3d          61: . . SEQUENCE
0105 31 0b          11: . . . SET
0107 30 09           9: . . . . SEQUENCE
0109 06 03           3: . . . . . OID 2.5.4.6: C
                      : 55 04 06
0114 13 02           2: . . . . . PrintableString  'US'
                      : 55 53
0118 31 0c          12: . . . SET
0120 30 0a          10: . . . . SEQUENCE
0122 06 03           3: . . . . . OID 2.5.4.10: O
                      : 55 04 0a
0127 13 03           3: . . . . . PrintableString  'gov'
                      : 67 6f 76
0132 31 0d          13: . . . SET
0134 30 0b          11: . . . . SEQUENCE
0136 06 03           3: . . . . . OID 2.5.4.11: OU
                      : 55 04 0b
0141 13 04           4: . . . . . PrintableString  'nist'
                      : 6e 69 73 74
0147 31 11          17: . . . SET
0149 30 0f          15: . . . . SEQUENCE
0151 06 03           3: . . . . . OID 2.5.4.3: CN
                      : 55 04 03
0156 13 08           8: . . . . . PrintableString  'Tim Polk'
                      : 54 69 6d 20 50 6f 6c 6b
0166 30 82 01 b4  436: . . SEQUENCE
0170 30 82 01 29  297: . . . SEQUENCE
0174 06 07          7: . . . . OID 1.2.840.10040.4.1: dsa
                      : 2a 86 48 ce 38 04 01
0183 30 82 01 1c  284: . . . . SEQUENCE
0187 02 81 80     128: . . . . . INTEGER
                      : d4 38 02 c5 35 7b d5 0b a1 7e 5d 72 59 63 55 d3
                      : 45 56 ea e2 25 1a 6b c5 a4 ab aa 0b d4 62 b4 d2
                      : 21 b1 95 a2 c6 01 c9 c3 fa 01 6f 79 86 83 3d 03
```

```
                    : 61 e1 f1 92 ac bc 03 4e 89 a3 c9 53 4a f7 e2 a6
                    : 48 cf 42 1e 21 b1 5c 2b 3a 7f ba be 6b 5a f7 0a
                    : 26 d8 8e 1b eb ec bf 1e 5a 3f 45 c0 bd 31 23 be
                    : 69 71 a7 c2 90 fe a5 d6 80 b5 24 dc 44 9c eb 4d
                    : f9 da f0 c8 e8 a2 4c 99 07 5c 8e 35 2b 7d 57 8d
0318 02 14           20: . . . . . INTEGER
                    : a7 83 9b f3 bd 2c 20 07 fc 4c e7 e8 9f f3 39 83
                    : 51 0d dc dd
0340 02 81 80       128: . . . . . INTEGER
                    : 0e 3b 46 31 8a 0a 58 86 40 84 e3 a1 22 0d 88 ca
                    : 90 88 57 64 9f 01 21 e0 15 05 94 24 82 e2 10 90
                    : d9 e1 4e 10 5c e7 54 6b d4 0c 2b 1b 59 0a a0 b5
                    : a1 7d b5 07 e3 65 7c ea 90 d8 8e 30 42 e4 85 bb
                    : ac fa 4e 76 4b 78 0e df 6c e5 a6 e1 bd 59 77 7d
                    : a6 97 59 c5 29 a7 b3 3f 95 3e 9d f1 59 2d f7 42
                    : 87 62 3f f1 b8 6f c7 3d 4b b8 8d 74 c4 ca 44 90
                    : cf 67 db de 14 60 97 4a d1 f7 6d 9e 09 94 c4 0d
0471 03 81 84       132: . . . BIT STRING  (0 unused bits)
                    : 02 81 80 a8 63 b1 60 70 94 7e 0b 86 08 93 0c 0d
                    : 08 12 4a 58 a9 af 9a 09 38 54 3b 46 82 fb 85 0d
                    : 18 8b 2a 77 f7 58 e8 f0 1d d2 18 df fe e7 e9 35
                    : c8 a6 1a db 8d 3d 3d f8 73 14 a9 0b 39 c7 95 f6
                    : 52 7d 2d 13 8c ae 03 29 3c 4e 8c b0 26 18 b6 d8
                    : 11 1f d4 12 0c 13 ce 3f f1 c7 05 4e df e1 fc 44
                    : fd 25 34 19 4a 81 0d dd 98 42 ac d3 b6 91 0c 7f
                    : 16 72 a3 a0 8a d7 01 7f fb 9c 93 e8 99 92 c8 42
                    : 47 c6 43
0606 a3 3e           62: . . [3]
0608 30 3c           60: . . . SEQUENCE
0610 30 19           25: . . . . SEQUENCE
0612 06 03            3: . . . . . OID 2.5.29.17: subjectAltName
                    : 55 1d 11
0617 04 12           18: . . . . . OCTET STRING
                    : 30 10 81 0e 77 70 6f 6c 6b 40 6e 69 73 74 2e 67
                    : 6f 76
0637 30 1f           31: . . . . SEQUENCE
0639 06 03            3: . . . . . OID 2.5.29.35: subjectAltName
                    : 55 1d 23
0644 04 18           24: . . . . . OCTET STRING
                    : 30 16 80 14 e7 26 c5 54 cd 5b a3 6f 35 68 95 aa
                    : d5 ff 1c 21 e4 22 75 d6
0670 30 09            9: . SEQUENCE
0672 06 07            7: . . OID 1.2.840.10040.4.3: dsa-with-sha
                    : 2a 86 48 ce 38 04 03
0681 03 2f           47: . BIT STRING  (0 unused bits)
                    : 30 2c 02 14 3c 02 e0 ab d9 5d 05 77 75 15 71 58
                    : 92 29 48 c4 1c 54 df fc 02 14 5b da 53 98 7f c5
                    : 33 df c6 09 b2 7a e3 6f 97 70 1e 14 ed 94
```

D.3 End-Entity Certificate Using RSA

This section contains an annotated hex dump of a 675 byte version 3 certificate. The certificate contains the following information: (a) the serial number is 256; (b) the certificate is signed with RSA and the MD2 hash algorithm; (c) the issuer's distinguished name is OU=Dept. Arquitectura de Computadors; O=Universitat Politecnica de Catalunya; C=ES (d) and the subject's distinguished name is CN=Francisco Jordan; OU=Dept. Arquitectura de Computadors; O=Universitat Politecnica de Catalunya; C=ES (e) the certificate was issued on May 21, 1996 and expired on May 21, 1997; (f) the certificate contains a 768 bit RSA public key; (g) the certificate is an end entity certificate (not a CA certificate); (h) the certificate includes an alternative subject name and an alternative issuer name - bothe are URLs; (i) the certificate include an authority key identifier and certificate policies extensions; and (j) the certificate includes a critical key usage extension specifying the public is intended for generation of digital signatures.

```
0000 30 80           : SEQUENCE   (size undefined)
0002 30 82 02 40  576: . SEQUENCE
0006 a0 03          3: . . [0]
0008 02 01          1: . . . INTEGER 2
                     : 02
0011 02 02          2: . . INTEGER 256
                     : 01 00
0015 30 0d         13: . . SEQUENCE
0017 06 09          9: . . . OID 1.2.840.113549.1.1.2:
                                  MD2WithRSAEncryption
                     : 2a 86 48 86 f7 0d 01 01 02
0028 05 00          0: . . . NULL
0030 30 68         88: . . SEQUENCE
0032 31 0b         11: . . . SET
0034 30 09          9: . . . . SEQUENCE
0036 06 03          3: . . . . . OID 2.5.4.6: C
                     : 55 04 06
0041 13 02          2: . . . . . PrintableString  'ES'
                     : 45 53
0045 31 2d         45: . . . SET
0047 30 2b         43: . . . . SEQUENCE
0049 06 03          3: . . . . . OID 2.5.4.10: O
                     : 55 04 0a
0054 13 24         36: . . . . . PrintableString
```

```
                    'Universitat Politecnica de Catalunya'
                    : 55 6e 69 76 65 72 73 69 74 61 74 20 50 6f 6c 69
                    : 74 65 63 6e 69 63 61 20 64 65 20 43 61 74 61 6c
                    : 75 6e 79 61
0092 31 2a         42: . . . SET
0094 30 28         40: . . . . SEQUENCE
0096 06 03          3: . . . . . OID 2.5.4.11: OU
                    : 55 04 0b
0101 13 21         33: . . . . . PrintableString
                    'OU=Dept. Arquitectura de Computadors'
                    : 44 65 70 74 2e 20 41 72 71 75 69 74 65 63 74 75
                    : 72 61 20 64 65 20 43 6f 6d 70 75 74 61 64 6f 72
                    : 73
0136 30 1e         30: . . SEQUENCE
0138 17 0d         13: . . . UTCTime  '960521095826Z'
                    : 39 36 30 37 32 32 31 37 33 38 30 32 5a
0153 17 0d         13: . . . UTCTime  '979521095826Z'
                    : 39 37 30 37 32 32 31 37 33 38 30 32 5a
0168 30 81 83     112: . . SEQUENCE
0171 31 0b         11: . . . SET
0173 30 09          9: . . . . SEQUENCE
0175 06 03          3: . . . . . OID 2.5.4.6: C
                    : 55 04 06
0180 13 02          2: . . . . . PrintableString  'ES'
                    : 45 53
0184 31 2d         12: . . . SET
0186 30 2b         16: . . . . SEQUENCE
0188 06 03          3: . . . . . OID 2.5.4.10: O
                    : 55 04 0a
0193 13 24         36: . . . . . PrintableString
                    'Universitat Politecnica de Catalunya'
                    : 55 6e 69 76 65 72 73 69 74 61 74 20 50 6f 6c 69
                    : 74 65 63 6e 69 63 61 20 64 65 20 43 61 74 61 6c
                    : 75 6e 79 61
0231 31 2a         42: . . . SET
0233 30 28         40: . . . . SEQUENCE
0235 06 03          3: . . . . . OID 2.5.4.11: OU
                    : 55 04 0b
0240 13 21         33: . . . . . PrintableString
                    'Dept. Arquitectura de Computadors'
                    : 44 65 70 74 2e 20 41 72 71 75 69 74 65 63 74 75
                    : 72 61 20 64 65 20 43 6f 6d 70 75 74 61 64 6f 72
                    : 73
0275 31 19         22: . . . SET
0277 30 17         20: . . . . SEQUENCE
0279 06 03          3: . . . . . OID 2.5.4.3: CN
                    : 55 04 03
0284 13 10         16: . . . . . PrintableString 'Francisco Jordan'
```

```
                    : 46 72 61 6e 63 69 73 63 6f 20 4a 6f 72 64 61 6e
0302 30 7c           2: . . SEQUENCE
0304 30 0d          13: . . . SEQUENCE
0306 06 09           9: . . . . OID 1.2.840.113549.1.1.1: RSAEncryption
                    : 2a 86 48 86 f7 0d 01 01 01
0317 05 00           0: . . . . NULL
0319 03 6b         107: . . . BIT STRING
                    : 00   (0 unused bits)
                    : 30 68 02 61 00 be aa 8b 77 54 a3 af ca 77 9f 2f
                    : b0 cf 43 88 ff a6 6d 79 55 5b 61 8c 68 ec 48 1e
                    : 8a 86 38 a4 fe 19 b8 62 17 1d 9d 0f 47 2c ff 63
                    : 8f 29 91 04 d1 52 bc 7f 67 b6 b2 8f 74 55 c1 33
                    : 21 6c 8f ab 01 95 24 c8 b2 73 93 9d 22 61 50 a9
                    : 35 fb 9d 57 50 32 ef 56 52 50 93 ab b1 88 94 78
                    : 56 15 c6 1c 8b 02 03 01 00 01
0428 a3 81 97      151: . . [3]
0431 30 3c          60: . . . SEQUENCE
0433 30 1f          31: . . . . SEQUENCE
0435 06 03           3: . . . . . OID 2.5.29.35: authorityKeyIdentifier
                    : 55 1d 23
0440 04 14          22: . . . . . OCTET STRING
                    : 30 12 80 10 0e 6b 3a bf 04 ea 04 c3 0e 6b 3a bf
                    : 04 ea 04 c3
0464 30 19          25: . . . . SEQUENCE
0466 06 03           3: . . . . . OID 2.5.29.15: keyUsage
                    : 55 1d 0f
0471 01 01           1: . . . . . TRUE
0474 04 04           4: . . . . . OCTET STRING
                    : 03 02 07 80
0480 30 19          25: . . . . SEQUENCE
0482 06 03           3: . . . . . OID 2.5.29.32: certificatePolicies
                    : 55 1d 20
0487 04 21          33: . . . . . OCTET STRING
                    : 30 1f 30 1d 06 04 2a 84 80 00 30 15 30 07 06 05
                    : 2a 84 80 00 01 30 0a 06 05 2a 84 80 00 02 02 01
                    : 0a
0522 30 1c          28: . . . . SEQUENCE
0524 06 03           3: . . . . . OID 2.5.29.17: subjectAltName
                    : 55 1d 11
0529 04 15          21: . . . . . OCTET STRING
                    : 30 13 86 11 68 74 74 70 3a 2f 2f 61 63 2e 75 70
                    : 63 2e 65 73 2f
0552 30 19          25: . . . . SEQUENCE
0554 06 03           3: . . . . . OID 2.5.29.18: issuerAltName
                    : 55 1d 12
0559 04 12          18: . . . . . OCTET STRING
                    : 30 14 86 12 68 74 74 70 3a 2f 2f 77 77 77 2e 75
                    : 70 63 2e 65
```

```
0579 30 80          : . SEQUENCE (indefinite length)
0581 06 07         7: . . OID
0583 05 00         0: . . NULL
0585 00 00         0: . . end of contents marker
0587 03 81 81     47: . BIT STRING
                   : 00      (0 unused bits)
                   : 5c 01 bd b5 41 88 87 7a 0e d3 0e 6b 3a bf 04 ea
                   : 04 cb 5f 61 72 3c a3 bd 78 f5 66 17 fe 37 3a ab
                   : eb 67 bf b7 da a8 38 f6 33 15 71 75 2f b9 8c 91
                   : a0 e4 87 ba 4b 43 a0 22 8f d3 a9 86 43 89 e6 50
                   : 5c 01 bd b5 41 88 87 7a 0e d3 0e 6b 3a bf 04 ea
                   : 04 cb 5f 61 72 3c a3 bd 78 f5 66 17 fe 37 3a ab
                   : eb 67 bf b7 da a8 38 f6 33 15 71 75 2f b9 8c 91
                   : a0 e4 87 ba 4b 43 a0 22 8f d3 a9 86 43 89 e6 50
0637 00 00         0: . . end of contents marker
```

D.4 Certificate Revocation List

This section contains an annotated hex dump of a version 2 CRL with one extension (cRLNumber). The CRL was issued by OU=nist;O=gov;C=us on July 7, 1996; the next scheduled issuance was August 7, 1996. The CRL includes one revoked certificates: serial number 18 (12 hex). The CRL itself is number 18, and it was signed with DSA and SHA-1.

```
0000 30 81 ba     186: SEQUENCE
0003 30 7c        124: . SEQUENCE
0005 02 01          1: . . INTEGER 1
                    : 01
0008 30 09          9: . . SEQUENCE
0010 06 07          7: . . . OID 1.2.840.10040.4.3: dsa-with-sha
                    : 2a 86 48 ce 38 04 03
0019 30 2a         42: . . SEQUENCE
0021 31 0b         11: . . . SET
0023 30 09          9: . . . . SEQUENCE
0025 06 03          3: . . . . . OID 2.5.4.6: C
                    : 55 04 06
0030 13 02          2: . . . . . PrintableString  'US'
                    : 55 53
0034 31 0c         12: . . . SET
0036 30 0a         10: . . . . SEQUENCE
0038 06 03          3: . . . . . OID 2.5.4.10: O
                    : 55 04 0a
0043 13 03          3: . . . . . PrintableString  'gov'
                    : 67 6f 76
0048 31 0d         13: . . . SET
0050 30 0b         11: . . . . SEQUENCE
0052 06 03          3: . . . . . OID 2.5.4.11: OU
                    : 55 04 0b
```

```
0057 13 04          4: . . . . . PrintableString  'nist'
                     : 6e 69 73 74
0063 17 0d         13: . . UTCTime  '970801000000Z'
                     : 39 37 30 38 30 31 30 30 30 30 30 30 5a
0078 17 0d         13: . . UTCTime  '970808000000Z'
                     : 39 37 30 38 30 38 30 30 30 30 30 30 5a
0093 30 22         34: . . SEQUENCE
0095 30 20         32: . . . SEQUENCE
0097 02 01          1: . . . . INTEGER 18
                     : 12
0100 17 0d         13: . . . . UTCTime  '970731000000Z'
                     : 39 37 30 37 33 31 30 30 30 30 30 30 5a
0115 30 0c         12: . . . . SEQUENCE
0117 30 0a         10: . . . . . SEQUENCE
0119 06 03          3: . . . . . . OID 2.5.29.21: reasonCode
                     : 55 1d 15
0124 04 03          3: . . . . . . OCTET STRING
                     : 0a 01 01
0129 30 09          9: . SEQUENCE
0131 06 07          7: . . OID 1.2.840.10040.4.3: dsa-with-sha
                     : 2a 86 48 ce 38 04 03
0140 03 2f         47: . BIT STRING  (0 unused bits)
                     : 30 2c 02 14 9e d8 6b c1 7d c2 c4 02 f5 17 84 f9
                     : 9f 46 7a ca cf b7 05 8a 02 14 9e 43 39 85 dc ea
                     : 14 13 72 93 54 5d 44 44 e5 05 fe 73 9a b2
```

Appendix E. Authors' Addresses

Russell Housley
SPYRUS
381 Elden Street
Suite 1120
Herndon, VA 20170
USA

EMail: housley@spyrus.com

Warwick Ford
VeriSign, Inc.
One Alewife Center
Cambridge, MA 02140
USA

EMail: wford@verisign.com

Tim Polk
NIST
Building 820, Room 426
Gaithersburg, MD 20899
USA

EMail: wpolk@nist.gov

David Solo
Citicorp
666 Fifth Ave, 3rd Floor
New York, NY 10103
USA

EMail: david.solo@citicorp.com

Appendix F. Full Copyright Statement