PKIX Working Group R. Housley (SPYRUS)
Internet Draft W. Ford (Verisign)
W. Polk (NIST)
D. Solo (BBN)
expires in six months October 14, 1997

Internet Public Key Infrastructure

X.509 Certificate and CRL Profile

<draft-ietf-pkix-ipki-part1-06.txt>

Status of this Memo

Abstract

This is the sixth draft of the Internet Public Key Infrastructure X.509 Certificate and CRL Profile. This draft is a complete specification. This text includes minor modifications over the previous draft. Please send comments on this document to the ietf-pkix@tandem.com mail list.

1 Executive Summary

This specification is one part of a multipart standard for the 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 structure 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 standards. In particular, this document's relationship with the IETF PEM specifications and the ISO 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 and ANSI extensions which may be useful in the Internet PKI and definition of new extensions to meet the Internet's requirements. The profiles are presented in the 1988 Abstract Syntax Notation One (ASN.1) rather than the 1993 syntax used in the ISO standards. The ASN.1 notation assumes implict tagging throughout.

This specification also includes path validation procedures in Section 6. These procedures are based upon the ISO definition, but the presentation assumes a self-signed root certificate. Implementations are required to derive the same results but are not required to use the specified procedures.

Finally, 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.

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 1993 syntax. Appendix B contains the same information in the 1993 ASN.1 notation. 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

Goal is to develop a profile and associated management structure to facilitate the adoption/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, as well as others. 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, as opposed to syntax.

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 certification practice Statement (CPS) generated by the 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 be more appropriate method 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 availablity. 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) are supported.

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 sophistication/attentiveness of the users themselves. This manifests itself in minimal user configuration responsibility (e.g., root 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 users, the Internet PKI profile is structured to support the individuals who generally operate Certification Authorities (CAs). Providing administrators with unbounded choices increases the chances that a subtle CA administrator mistake will result in broad compromise. Also, unbounded choices greatly complicates the software that must 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.

```
      +---+
      | C |                       +------------+
      | e | <-------------------->| End entity |
      | r |       Operational     +------------+
      | t |       transactions         ^
      |   |      and management        |  Management
      | / |       transactions         |  transactions
      |   |                            |
      | C |    PKI users               v
      | R |             -------+-------+--------+------
      | L |   PKI management   ^                ^
      |   |      entities      |                |
      |   |                    v                |
      | R |                 +------+            |
      | e | <-------------- | RA   | <-----+    |
      | p |   certificate   |      |       |    |
      | o |       publish   +------+       |    |
      | s |                                |    |
      | I |                                v    v
      | t |                            +------------+
      | o | <--------------------------|     CA     |
      | r |   certificate publish      +------------+
      | y |           CRL publish             ^
      |   |                                   |
      +---+                                   |    Management
                                              |    transactions
                                              v
                                          +------+
                                          |  CA  |
                                          +------+
```

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

Application of public key technology requires the user of a public key to be confident that the public key belongs to 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 achieved by having a trusted certification authority (CA) digitally sign each certificate. 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.

The standard known as ITU-T X.509 (formerly CCITT X.509) or ISO/IEC 9594-8, which was first published in 1988 as part of the X.500 Directory recommendations, defines a standard certificate format. 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 are used to support directory access control.

The Internet Privacy Enhanced Mail (PEM) proposals, 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 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-AM].

ISO/IEC and ANSI X9 have also developed standard extensions for use in the v3 extensions field [X.509-AM][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 and ANSI 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, 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 must 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 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 was based upon 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 the root;

(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 many environments, the local domain is often the most trusted. Initialization and key-pair-update operations can often be more effectively conducted between an end entity and a local management system.

(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 the full process of certificate chain processing to be implemented in software.

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 might 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). Entries are added to CRLs as revocations occur, and an entry may be removed when the certificate expiration date is reached.

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.

Another potential problem with CRLs is the risk of a CRL growing to an entirely unacceptable size. In the 1988 and 1993 versions of X.509, the CRL for the end-user certificates needed to cover the entire population of end-users for one CA. It is desirable to allow such populations to be in the range of thousands, tens of thousands,

or possibly even hundreds of thousands of users. The end-user CRL is therefore at risk of growing to such sizes, which present major communication and storage overhead problems. With the version 2 CRL format, introduced along with the v3 certificate format, it becomes possible to arbitrarily divide the population of certificates for one CA into a number of partitions, each partition being associated with one CRL distribution point (e.g., directory entry or URL) from which CRLs are distributed. Therefore, the maximum CRL size can be controlled by a CA. Separate CRL distribution points can also exist for different revocation reasons. For example, routine revocations (e.g., name change) may be placed on a different CRL to revocations resulting from suspected key compromises, and policy may specify that the latter CRL be updated and issued more frequently than the former.

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.

Furthermore, it is recognized that on-line methods of revocation notification may be applicable in some environments as an alternative to the X.509 CRL. On-line revocation checking significantly reduces the latency between a revocation report and the next issue of a CRL. 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 must trust the on-line validation service while the repository did not need to be trusted.

Therefore, this profile also considers standard approaches to on-line revocation notification. The PKIX series of specifications defines a set of standard message formats supporting these functions in [PKIXOCSP]. The protocols for conveying these messages in different environments are also specified.

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 request/delivery procedures based on E-mail, http, X.500, and WHOIS++. These specifications include definitions of, and/or references to, message formats and procedures for supporting all of the above operational environments, including definitions of or references to appropriate MIME content types.

Operational protocols supporting these functions are defined in the

PKIX specifications [PKIXLDAP], [PKIXFTP] and [PKIXOCSP].

3.5 Management Protocols

Management protocols are required to support on-line interactions between Public Key Infrastructure (PKI) components. For example, 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 in it necessary 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 of a CA, 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 the information necessary to establish cross-certificates between those CAs.

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 defines a set of standard message formats supporting the above functions in [PKIXMGMT]. The protocols for conveying these messages in different environments (on-line, e-mail, and WWW) are also specified in [PKIXMGMT].

4 Certificate and Certificate Extensions Profile

This section presents a profile for public key certificates that will foster interoperability and a reusable public key infrastructure. This section is based upon the X.509 V3 certificate format [COR95][X.509-AM] and the standard certificate extensions defined in the Amendment [X.509-AM]. The ISO 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 public key infrastructure 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. Other efforts are looking at certificate profiles for payment systems.

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,
     signature            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 must be v2 or v3
        subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                             -- If present, version must be v2 or v3
        extensions      [3]  EXPLICIT Extensions OPTIONAL
                             -- If present, version must be v3
        }

   Version  ::=  INTEGER  {  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 a proposed use of the X.509 v3 certificate for the Internet.

4.1.1 Certificate Fields

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

4.1.1.1 tbsCertificate

The first field in the sequence is the tbsCertificate. This is a itself a sequence, and contains the names of the subject and issuer, a public key associated with the subject an expiration date, and other associated information. The fields of the basic tbsCertificate 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 algorithm identifier for the 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  }
```

and it is used to identify a cryptographic algorithm. The OBJECT IDENTIFIER algorithm identifies the algorithm (such as RSA with SHA-1). The contents of the optional parameters field will vary according to the algorithm identfied and the purpose of the algorithm identifier.

In this case, the parameters field will usually be empty. Section 7.2 lists the supported algorithms for this specification and describes the contents of the parameters fields for each algorithm.

This field should contain the same algorithm identifier as the signature field in the sequence tbsCertificate (see section 4.1.2.3)

4.1.1.3 signature

The signature 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 a one-way hash function. The one-way hash function output value is encrypted (e.g., using RSA Encryption) to form the signed quantity. 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 tbscertificate. 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 is a sequence which 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, an expiration date, a version number and a serial number; some will 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 shall 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 certification authority 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. Section 7.2 lists the supported signature algorithms.

This field should contain the same algorithm identifier as the signatureAlgorithm field in the sequence Certificate (see section 4.1.1.2).

4.1.2.4 Issuer Name

The issuer name identifies the entity who has signed (and issued the certificate). The issuer identity may be carried in the issuer name field and/or the issuerAltName extension. If identity information is present only in the issuerAltName extension, then the issuer name may be an empty sequence and the issuerAltName extension must be critical.

Where it is non-null, the issuer name field shall contain an X.500 distinguished name (DN). The issuer field is defined as the X.501 type Name. 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

-- Directory string type --

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

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 is defined as a choice of PrintableString, TeletexString, BMPString and UniversalString. Conforming CAs shall choose from these options as follows:

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

(b) failing (a), if the teletexString character set is sufficient, the string will be represented as a TeletexString;

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

(d) failing (a), (b) and (c), the string shall be represented as a UniversalString.

Standard sets of attributes have been defined in the X.500 series of specifications. Where CAs issue certificates with X.501 type names, it is recommended that these attributes types be used.

4.1.2.5 Validity

This field indicates the period of validity of the certificate, and consists of two dates, the first and last on which the certificate is valid. The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate, i.e. publish revocation data. 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 shall always encode certificate validity dates through the year 2049 as UTCTime; certificate validity dates in 2050 or later shall 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 shall be expressed Greenwich Mean Time (Zulu) and shall include seconds (i.e., times are YYMMDDHHMMSSZ), even where the number of seconds is zero. Conforming systems shall 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 shall be expressed Greenwich Mean Time (Zulu) and shall include seconds (i.e., times are YYYYMMDDHHMMSSZ), even where the number of seconds is zero. GeneralizedTime values shall not include fractional seconds.

4.1.2.6 Subject Name

The subject name identifies the entity associated with the public key stored in the subject public key field. The subject identity may be carried in the subject field and/or the subjectAltName extension. If identity information is present only in the subjectAltName extension (e.g., a key bound only to an email address or URI), then the subject name may be an empty sequence and the subjectAltName extension must be critical.

Where it is non-null, the subject name field shall 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, and shall follow the encoding rules for the issuer name field (see 4.1.2.4).

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

The subject and issuer unique identifier 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 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 number is 3 (see 4.1.2.x). 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 Certificate Extensions

The extensions defined for X.509 v3 certificates provide methods for associating additional attributes with users or public keys, for managing the certification hierarchy, and for managing CRL distribution. 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 (an application validating a certificate) must reject the certificate if it encounters a critical extension it does not recognize. A non-critical extension may be ignored if it is not recognized. The following presents 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 be used in a general context.

Each extension includes an object identifier and an ASN.1 structure. When an extension appears in a certificate, the object identifier 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 (4.2.1.1). An extension may also include the optional boolean critical; critical's default value is FALSE. The text for each extension specifies the acceptable values for the critical field.

Conforming CAs are required to support the basic Constraints extension (Section 4.2.1.10), the key usage extension (4.2.1.3) and certificate policies extension (4.2.1.5). If the CA issues certificates with an empty sequence for the subject field, the CA must support the subjectAltName extension. If the CA issues certificates with an empty sequence for the issuer field, the CA must support the issuerAltName extension. 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 shall recognize extensions which shall or may be critical. These extensions are: key usage (4.2.1.3), certificate policies (4.2.1.5), the alternative subject name (4.2.1.7), issuer alternative name (4.2.1.8), basic constraints (4.2.1.10), name constraints (4.2.1.11), policy constraints (4.2.1.12), and extended key usage (4.2.1.14).

In addition, this profile recommends support for key identifiers (4.2.1.1 and 4.2.1.2), CRL distribution points (4.2.1.13), and authority information access (4.2.2.1).

4.2.1 Standard Extensions

This section identifies standard certificate extensions defined in [X.509-AM] for use in the Internet Public Key Infrastructure. Each extension is associated with an object identifier defined in [X.509-AM]. These object identifiers are members of the certificateExtension arc, which is defined by the following:

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

4.2.1.1 Authority Key Identifier

The authority key identifier extension provides a means of identifying the particular public key used to sign a certificate. This extension would be used where an issuer has multiple signing keys (either due to multiple concurrent key pairs or due to changeover). In general, this extension should be included in certificates.

The identification can 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 key identifier method is recommended in this profile. Conforming CAs that generate this extension shall include or omit both authorityCertIssuer and authorityCertSerialNumber. If authorityCertIssuer and authorityCertSerialNumber are omitted, the keyIdentifier field shall be present.

This extension shall 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 the particular public key used in an application. Where a reference to a public key identifier is needed (as with an Authority Key Identifier) and one is not included in the associated certificate, a SHA-1 hash of the subject public key shall be used. The hash shall be calculated over the value (excluding tag and length) of the subject public key field in the certificate. This extension should be marked non-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 nonRepudiation bits would be asserted. Likewise, when an RSA key should be used only for key management, the keyEncipherment bit would be asserted. The profile recommends that when used, this be marked as a critical extension.

```
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 to verifying digital signatures that have purposes other than non-repudiation, certificate signature, and CRL signature. For example, The digitalSignature bit is asserted when the subject public key is used to provide authentication.

The nonRepudiation bit is asserted when the subject public key is used to verifying 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 exclusively for key management, then this bit must 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 exclusively for key management, then this bit must 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 CRLs. This bit may only be asserted in CA certificates.

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 encipherOnly 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. The meaning of the decipherOnly bit is undefined in the absence of the keyAgreement bit.

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

4.2.1.4 Private Key Usage Period

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 shall not generate certificates with private key usage period extensions unless at least one of the two components is present.

This profile recommends against the use of this extension. CAs conforming to this profile shall not generate certificates with critical private key usage period extensions. Where used, notBefore and notAfter are represented as GeneralizedTime and shall 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. This profile strongly recommends that a simple OID be present in this field. Optional qualifiers which may be present are expected to provide information about obtaining CA rules, not 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, or must reject the certificate. (Applications without specific policy requirements are not required to list acceptable

policies, and may accept any valid certificate regardless of policy even if the extension is marked critical.)

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

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. It is recommended that only the lowest-level certificate issued by one organization in a certification path contain a user notice.

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 ::= { id-pkix 2 }  -- pkix arc for qualifier types
   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  IA5String,
     noticeNumbers SEQUENCE OF INTEGER }

   DisplayText ::= CHOICE {
     visibleString VisibleString,
     bmpString     BMPString }
```

4.2.1.6 Policy Mappings

This extension is used in CA certificates. It lists one or more pairs of object identifiers; 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 is always 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 rfc822 name (electronic mail address), a DNS name, an IP address, and a URI. Other options exist, including completely local definitions. Multiple instances of a name and multiple name forms may be included. Whenever such identities are to be bound into a certificate, the subject alternative name (or issuer alternative name) extension shall be used. (Note: a form of such an identifier may also be present in the subject distinguished name; however, the alternative name extension is the preferred location for finding such information.)

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 shall be empty (an empty sequence), and the subjectAltName extension shall be present. If the subject field contains an empty sequence, the subjectAltName extension shall be marked critical.

Where the subjectAltName extension contains a dNSName, this name may contain the wildcard character. An "*" is the wildcard character. Where a dNSName includes a wildcard, the subject of this certificate is a subnet or a collection of hosts. Examples include *.bar.com and www*.bar.com.

Where the subjectAltName extension contains an rfc822Name, this name may also include the wildcard character. Use of the wildcard is limited to the host name.

Where the subjectAltName extension contains a uniformResourceIdentifier, the URI is a pointer to a sequence of certificates issued by this CA (and optionally other CAs) to this subject. The URI may not contain the wildcard character in the host name.

The URI must be an absolute, not relative, pathname and must specify the host. This specification recognizes the following values for the URI scheme: ftp, http, ldap, and mailto. The mailto scheme indicates that mail sent to the specified address will generate an electronic mail response (to the sender) containing the subject's certificates. No message is required. If the URI scheme is ftp, then the information is available through anonymous ftp. If the URI scheme is http or ldap, then the information may be retrieved using that protocol.

(If the URI specifies any other scheme, contains a relative pathname, or omits the host, the semantics are not defined by this specification.)

When the subjectAltName extension contains a iPAddress, the address shall be stored in the octet string in "network byte order," as specified in RFC791. 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.

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 shall 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.

```
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 Name

As with 4.2.1.7, this extension is used to associate Internet style identities with the certificate issuer. If the only issuer identity included in the certificate is an alternative name form (e.g., an electronic mail address), then the issuer distinguished name shall be empty (an empty sequence), and the issuerAltName extension shall be present. If the subject field contains an empty sequence, the issuerAltName extension shall be marked critical.

Where the issuerAltName extension contains a URI, the following semantics shall be assumed: the URI is a pointer to an ASN.1 sequence of certificates issued to this CA (and optionally other CAs). 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.

Where the issuerAltName extension contains a dNSName, rfc822Name, or a URI, wildcard characters are not permitted.

```
     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 is always 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 profile requires the use of this extension, and it shall always be critical for CA 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 shall be used only in a CA certificate, indicates a name space within which all subject names in subsequent certificates in a certification path must be located. Restrictions may apply to the subject distinguished name or subject alternative names. 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.

Restrictions for the rfc822, dNSName, and uri name forms are all expressed in terms of strings with wild card matching. An "*" is the wildcard character. For uris and rfc822 names, the restriction applies to the host part of the name. Examples would be foo.bar.com; www*.bar.com; *.xyz.com.

Legacy implementations exist where an RFC 822 name is embeded in the subject distinguished name as a PKCS #9 e-mail attribute, which has the ASN.1 type EmailAddress. When rfc822 names are constrained, but the certificate does not include a subject alternative name, the

rfc822 name constraint must be applied to PKCS #9 e-mail attributes in the subject distinguished name. The ASN.1 syntax for EmailAddress and the corresponding OID are supplied below.

```
EmailAddress ::= IA5String

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

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

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

The syntax and semantics for name constraints for otherName, ediPartyName, iPAddress, 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 must 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 shall 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 }

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

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

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

4.2.1.13 CRL Distribution Points

The CRL distribution points extension identifies how CRL information is obtained. The extension shall 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 shall 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 shall include revocations for all reasons. If the distributionPoint omits cRLIssuer, the CRL shall 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) }
```

4.2.1.14 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 ITU-T Rec. X.660 | ISO/IEC 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 shall be used only for one of the purposes indicated.

If the extension is flagged non-critical, then it indicates the intended purpose or purposes of the key, and may be used in finding

the correct key/certificate of an entity that has multiple keys/certificates. It is an advisory field and does not imply that usage of the key is restricted by the certification authority to the purpose indicated. (Using applications may nevertheless require that a particular purpose be indicated in order for the certificate to be acceptable to that application.)

If a certificate contains both a critical key usage field and a critical extended key usage field, then both fields shall be processed independently and the certificate shall only be used for a purpose consistent with both fields. If there is no purpose consistent with both fields, then the certificate shall 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-ipsecEndSystem          OBJECT IDENTIFIER ::=   {id-kp 5}
-- IP security end system (host or router)
-- Key usage bits that may be consistent: digitalSignature and/or
--                         (keyEncipherment or keyAgreement)
--
id-kp-ipsecTunnel             OBJECT IDENTIFIER ::=   {id-kp 6}
-- IP security tunnel termination
-- Key usage bits that may be consistent: digitalSignature and/or
--                         (keyEncipherment or keyAgreement)
--
```

```
id-kp-ipsecUser               OBJECT IDENTIFIER ::=   {id-kp 7}
-- IP security user
-- Key usage bits that may be consistent: digitalSignature 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
```

4.2.2 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 uder 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 is always 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-ocsp OBJECT IDENTIFIER ::= { id-ad 1 }

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.

This profile defines an object identifier for the On-line Certificate Status Protocol (OCSP) that will be defined in [PKIXOCSP]. When id-ad-ocsp appears as accessMethod, the accessLocation field describes the on-line status server and the access protocol to obtain current certificate status information for the certificate containing this extension.

This profile defines an object identifier to obtain a description of the 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. The syntax of the referenced CA Issuers description will be defined in [PKIXOCSP]. When id-ad-caIssuers appears as accessMethod, the accessLocation field describes the referenced description server and the access protocol to obtain referenced description.

Additional access descriptors will likely be defined in the future.

The authorityInfoAccess extension may be included in a PKCS 7 encapsulation as an X.501 ATTRIBUTE. This attribute can then be used to locate certificates automatically rather than include the certificates directly. The intended effect is to reduce the size of the encapsulated message or object.

PKCS 9 identifies attributes for inclusion in PKCS 7, referencing X.520 standard attributes and defining additional attributes unique to PKCS 9. The attributes defined in X.520 are based on the definition of ATTRIBUTE in ITU-T X.501 | ISO/IEC 9594-2.

The following syntax defines authorityInfoAccess as an ATTRIBUTE suitable for inclusion in a PKCS #7 message:

```
authorityInfoAccess ATTRIBUTE ::= {
```

```
    WITH SYNTAX    authorityInfoAccessSyntax,
    ID             id-pe-authorityInfoAccess }
```

Other parts of the PKIX specifications [PKIXOCSP] [PKIXLDAP] establish requirements on certificate retrieval mechanisms. It is expected that applications using the URI form of the authorityInfo field for such a purpose will:

1. Prepend a suitable HTTP retrieval primitive to the URL (e.g. "GET").

2. Append a filename to the URL.

3. Use the result to retrieve a file containing the requested certificate.

4. Use the authorityInfoAccess extension in that and subsequent certificates to complete a certificate path.

The filename will be formed as the IA5string representation of SHA1(Issuer DN | certificate serial number) concatenated with ".cer." The IA5String representation will display the SHA1 result as a hexidecimal number using digits and the lowercase letters 'a' through 'f.' The SignerInfo syntax of PKCS 7 provides the necessary information as issuerAndSerialNumber.

The specified file will contain a single DER encoded certficate.

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. Emphasis is placed on support for X.509 v2 CRLs. 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, and 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 status mechanisms are provided. Conforming CAs that issue CRLs are required to issue version 2 CRLs, and must include the date by which the next CRL will be issued in the nextUpdate field (Section 5.1.2.5). 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,
     signature            BIT STRING  }

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

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

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
```

The following items describe the proposed 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 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. Section 7.2 lists the supported signature algorithms. Conforming CAs shall use the algorithm identifiers presented in Section 7.2 when signing with a supported signature algorithm.

5.1.1.3 signature

The signature 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 a one-way hash function. The one-way hash function output value is encrypted (e.g., using RSA Encryption) to form the signed quantity. This signature value is then ASN.1 encoded as a BIT STRING and included in the CRL's signature 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 special case where a CA has not revoked any unexpired certificates it has issued. It is expected that nearly all CRLs issued in the

Internet PKI will contain one or more lists of revoked certificates. Similarly, 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 expected in this profile, this field shall be present and shall specify version 2 (the integer value is 1). If neither CRL extensions nor CRL entry extensions are present, version 1 CRLs are recommended. In this case, the field shall be ommitted.

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.

5.1.2.3 Issuer Name

The issuer name identifies the entity who has signed (and issued the CRL). The issuer identity may be carried in the issuer name field and/or the issuerAltName extension. If identity information is present only in the issuerAltName extension, then the issuer name may be an empty sequence and the issuerAltName extension must be critical.

Where it is non-null, the issuer name field shall contain an X.500 distinguished name (DN). The issuer name field is defined as the X.501 type Name, and shall follow the encoding rules for the issuer name field in the certificate (see 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 shall encode thisUpdate as UTCTime for dates through the year 2049. CAs conforming to this profile that issue CRLs shall encode thisUpdate as GeneralizedTime for dates in the year 2050 or later.

Where encoded as UTCTime, thisUpdate shall be specified and interpreted as defined in Section 4.1.2.5.1. Where encoded as GeneralizedTime, thisUpdate shall 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. 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-AM]. The behavior of clients processing CRLs which omit nextUpdate is not specified by this profile.

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

Where encoded as UTCTime, nextUpdate shall be specified and interpreted as defined in Section 4.1.2.5.1. Where encoded as GeneralizedTime, nextUpdate shall 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 are uniquely identified by the combination of the issuer name or issuer alternative name along with the user certificate serial number. The date on which the revocation occurred is specified. The time for revocationDate shall 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 number is 2 (see 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 for X.509 v2 CRLs [X.509-AM] [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 an critical

extension which it does not know how to process. However, an unrecognized non-critical extension may be ignored. The following presents 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 support the CRL number extension (5.2.3), and include it in all CRLs issued. Conforming applications are required to support the critical and optionally critical CRL extensions issuer alternative name (5.2.2), issuing distribution point (5.2.4) and delta CRL indicator (5.2.5).

5.2.1 Authority Key Identifier

The authority key identifier extension provides a means of identifying the particular public 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. The key identifier method is recommended in this profile. This extension would be used where an issuer has multiple signing keys, either due to multiple concurrent key pairs or due to changeover. In general, this non-critical extension should be included in certificates.

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 shall be used.

Further, if the only issuer identity included in the CRL is an alternative name form (e.g., an electronic mail address), then the issuer distinguished name should be empty (an empty sequence), the issuerAltName extension should be used, and the issuerAltName extension must be marked critical.

The object identifier 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 given CA through a specific CA X.500 Directory entry or CRL distribution point. This extension allows users to easily determine when a particular CRL supersedes another CRL. CAs conforming to this profile shall include this extension in all CRLs.

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

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

5.2.4 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. Since this extension is critical, all certificate users must be prepared to receive CRLs with 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.

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) shall appear in one distribution point, and the revocations with other reason codes shall appear in another distribution point. The reason codes associated with a distribution point must be specified in onlySomeReasons. If onlySomeReasons does not appear, the distribution point must contain revocations for all reason codes.

Where the issuingDistributionPoint extension contains a URL, the following semantics shall 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 }
```

5.2.5 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 shall 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 shall not be issued without a corresponding CRL. The value of CRLNumber for both the delta-CRL and the corresponding CRL shall be identical.

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

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

deltaCRLIndicator ::= BaseCRLNumber

BaseCRLNumber ::= CRLNumber
```

5.2.6 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. Conforming applications if an implementation ignored this extension it could not correctly attribute CRL entries to certificates.

5.3 CRL Entry Extensions

The CRL entry extensions already defined by ANSI X9 and ISO for X.509 v2 CRLs [X.509-AM] [X9.55] provide methods for associating additional attributes with CRL entries. 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 presents 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 are non-critical; support for these extensions is optional for conforming CAs and applications. However, CAs that issue CRLs are strongly encouraged to include reason codes (5.3.1) 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 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 shall 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 a id-holdinstruction-callissuer must call the certificate issuer or reject the certificate. Conforming applications which encounter a id-holdinstruction-reject ID shall reject the transaction. id-holdinstruction-none is semantically equivalent to the absence of a holdInstructionCode. 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, but it must be later than the issue date of the previously issued CRL. Remember that the revocation date in the CRL entry specifies the date that the CA revoked the certificate. Whenever this information is available, CAs are strongly encouraged to share it with CRL users.

The GeneralizedTime values included in this field shall be expressed in Greenwich Mean Time (Zulu), and shall be specified and interpreted

as defined in Section 4.1.2.5.2.

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

invalidityDate ::=  GeneralizedTime
```

6 Certificate Path Validation

Certification path validation procedures for the Internet PKI are based on Section 12.4.3 of [X.509-AM]. Certification path processing verifies the binding between the subject distinguished 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 shall be functionally equivalent to the external behaviour resulting from this procedure. Any algorithm may be used by a particular implementation so long as it derives the correct result.

The following text assumes that all valid paths begin with the public key of a single "most-trusted CA". 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."

The text assumes that this public key 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 public key it contains 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 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; and

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

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 set of initial policy identifiers.

(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 (f) through (i) 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 is not expired, and (if present) the private key usage period is satisfied,

(3) the certificate has not been revoked (this may be determined by obtaining current CRL, current status information, or by out-of-band mechanisms), and

(4) the subject and issuer names chain correctly. (If the certificate has an empty sequence in the name field, name chaining will use the critical subjectAltNames and issuerAltNames fields.)

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

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

(d) Verify that policy information is consistent:

(1) if the explicit policy state variable is less than or equal to i, an appropriate policy identifier must appear in the certificate; and
(2) if the policy mapping variable is less than or equal to i, the policy identifier may not be mapped.

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

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

(g) 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.

(h) 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.

(i) 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 (a) its current value and (b) 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 (a) its current value and (b) the sum of q and i (the current certificate in the sequence).

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.

Notes: It is possible to specify an extended version of the above certification path processing procedure which results in default behaviour identical to the rules of Privacy Enhanced Mail [RFC 1422]. In this extended version, additional inputs to the procedure are a list of one or more Policy Certification Authoritys (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.

This procedure may also be extended 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. These self-signed certificates permit the path validation module to automatically incorporate local security policy and requirements.

7 Algorithm Support

This section describes cryptographic algorithms which may be used with this standard. The section describes one-way hash functions and digital signature algorithms which may be used to sign certificates and CRLs, and identifies object identifiers 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, this profile requires conforming CAs and applications to conform when they use the algorithms identified here.

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, but 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 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.

<< More information on the attack and its implications can be

obtained from a RSA Laboratories security bulletin. These bulletins are available from <http://www.rsa.com/>. >>

7.1.2 MD5 One-way Hash Function

MD5 was developed by Ron Rivest in 1991. The algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. The MD5 message digest algorithm is specified by RFC 1321, "The MD5 Message-Digest Algorithm"[RFC1321].

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.

7.1.2 SHA-1 One-way Hash Function

SHA-1 was developed by the U.S. Government. The algorithm takes as input a message of arbitrary length and produces as output 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 Section 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 algorithmidentifier which appears in the signatureAlgorithm field in a Certificate or CertificateList. This algorithmidentfier 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 object identifier, and may include associated parameters. This section identifies OIDS for RSA and DSA and the corresponding parameters.

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 [PKCS#1]. As defined in PKCS #1, the ASN.1 object identifier 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 [PKCS#1]. As defined in PKCS #1, the ASN.1 object identifier 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 defined in by the OSI Interoperability Workshop in [OIW]. Padding conventions described in PKCS #1, section 8.1, must be used. As defined in [OIW], the ASN.1 object identifier used to identify this signature algorithm is:

```
sha1WithRSASignature OBJECT IDENTIFIER  ::=  {
    iso(1) identified-organization(3) oiw(14)
    secsig(3) algorithm(2) 29  }
```

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

The data to be signed (e.g., the one-way hash function output value)

is first ASN.1 encoded as an OCTET STRING and the result is encrypted (e.g., using RSA Encryption) to form the signed quantity. When signing, the RSA algorithm generates an integer y. This signature value is then ASN.1 encoded as a BIT STRING, such that the most significant bit in y is the first bit in the bit string and the least significant bit in y is the last bit in the bit string, and included in the Certificate or CertificateList (in the signature field).

(In general the conversion to a bit string occurs in two steps. The integer y is converted to an octet string such that the first octet has the most significance and the last octet has the least significance. The octet string is converted into a bit string such that the most significant bit of the first octet shall become the first bit in the bit string, and the least significant bit of the last octet is the last bit in the BIT STRING.)

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 object identifiers 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 }
```

The id-dsa-with-sha1 algorithm syntax has NULL parameters. The DSA parameters in the subjectPublicKeyInfo field of the certificate of the issuer shall apply to the verification of the signature.

If the subjectPublicKeyInfo AlgorithmIdentifier field has NULL parameters and the CA signed the subject certificate using DSA, then the certificate issuer's parameters apply to the subject's DSA key. If the subjectPublicKeyInfo AlgorithmIdentifier field has NULL parameters and the CA signed the subject with a signature algorithm other than DSA, then clients shall not validate the certificate.

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 standard may convey a public key for any public key algorithm. The certificate indicates the algorithm through an algorithmidentifier. This algorithm identfieier 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 object identifier 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 object identifier 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 subjectPubliKey.

This object identifier 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 Section 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

This diffie-hellman object identifier supported by this standard is defined by ANSI X9.42.

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

The dhpublicnumber object identifier 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, would have ASN.1 type DHParameter for this algorithm.

```
DHParameter ::= SEQUENCE {
  prime INTEGER, -- p
  base INTEGER, -- g }
```

The fields of type DHParameter have the following meanings:

prime is the prime p.

base is the base g.

The Diffie-Hellman public key (an INTEGER) is mapped to a

subjectPublicKey (a BIT STRING) as follows: the most significant bit (MSB) of the INTEGER becomes the MSB of the BIT STRING; the least significant bit (LSB) of the INTEGER becomes the LSB of the BIT STRING.

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 object identifier supported by this standard 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. 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. The parameters are included using the following ASN.1 structure:

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

If the subjectPublicKeyInfo AlgorithmIdentifier field has NULL parameters and the CA signed the subject certificate using DSA, then the certificate issuer's parameters apply to the subject's DSA key. If the subjectPublicKeyInfo AlgorithmIdentifier field has NULL parameters and the CA signed the subject with a signature algorithm other than DSA, then clients shall not validate 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 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.

References

[COR95] ISO/IEC JTC 1/SC 21, Technical Corrigendum 2 to ISO/IEC 9594-8: 1990 & 1993 (1995:E), July 1995.

[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.

[OIW] Stable Implementation Agreements for Open Systems Interconnection Protocols: Part 12 - OS Security, Output from the June 1995 Open Systems Environment

Implementors' Workshop (OIW).

[PKCS#1] PKCS #1: RSA Encryption Standard, Version 1.4, RSA Data Security, Inc., 3 June 1991.

[RC95] Rogier, N. and Chauvaud, P., "The compression function of MD2 is not collision free," Presented at Selected Areas in Cryptography '95, Carleton University, Ottawa, Canada, 18-19 May 1995.

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

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

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

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

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

[RFC 1777] W. Yeong, T. Howes & S. Kille, "Lightweight Directory Access Protocol," March 1995.

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

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

[RFC 1959] T. Howes, M. Smith, "An LDAP URL Format", RFC 1959, June 1996.

[PKIXMGMT] C. Adams, S. Farrell, "Internet Public Key Infrastructure Certificate Management Protocols", draft-ietf-pkix-ipki3cmp-04.txt, September 1997

[PKIXLDAP] S. Boyeun, T. Howes and P. Richard "Internet Public Key Infrastructure Operational Protocols - LDAP", draft-ietf-pkix-ipki2opp-03.txt, September 1997.

[PKIXOCSP] M. Myers, in "Internet Public Key Infrastructure Part 2: Operational Protocols", draft-ietf-pkix-ipki2opp-02.txt,

July 1997.

[PKIXFTP] R. Housley, "Internet Public Key Infrastructure Operational Protocols: FTP and HTTP", draft-ietf-pkix-opp-ftp-http-00.txt, September 1997.

[SDN.701R] SDN.701, "Message Security Protocol", Revision 4.0 1996-06-07 with "Corrections to Message Security Protocol, SDN.701, Rev 4.0, 96-06-07." August 30, 1996.

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

[X.509-AM] ISO/IEC JTC1/SC 21, Draft Amendments DAM 4 to ISO/IEC 9594-2, DAM 2 to ISO/IEC 9594-6, DAM 1 to ISO/IEC 9594-7, and DAM 1 to ISO/IEC 9594-8 on Certificate Extensions, 1 December, 1996.

[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.

Patent Statements

The Internet PKI relies on the use of patented public key technology and secure hash technology for digital signature services. This specification also references public key encryption technology for provisioning key exchange services.

The Internet Standards Process as defined in RFC 1310 requires a written statement from the Patent holder that a license will be made available to applicants under reasonable terms and conditions prior to approving a specification as a Proposed, Draft or Internet Standard.

Patent statements for DSA, RSA, and Diffie-Hellman follow. These statements have been supplied by the patent holders, not the authors of this profile.

The Internet Society, Internet Architecture Board, Internet Engineering Steering Group and the Corporation for National Research Initiatives take no position on the validity or scope of the following patents and patent applications, nor on the appropriateness of the terms of the assurance. The Internet Society and other groups

mentioned above have not made any determination as to any other intellectual property rights which may apply to the practice of this standard. Any further consideration of these matters is the user's own responsibility.

Digital Signature Algorithm (DSA)

The U.S. Government holds patent 5,231,668 on the Digital Signature Algorithm (DSA), which has been incorporated into Federal Information Processing Standard (FIPS) 186. The patent was issued on July 27, 1993.

The National Institute of Standards and Technology (NIST) has a long tradition of supplying U.S. Government-developed techniques to committees and working groups for inclusion into standards on a royalty-free basis. NIST has made the DSA patent available royalty-free to users worldwide.

Regarding patent infringement, FIPS 186 summarizes our position; the Department of Commerce is not aware of any patents that would be infringed by the DSA. Questions regarding this matter may be directed to the Deputy Chief Counsel for NIST.

RSA Signature and Encryption

The Massachusetts Institute of Technology has granted RSA Data Security, Inc., exclusive sub-licensing rights to the following patent issued in the United States:

Cryptographic Communications System and Method ("RSA"), No. 4,405,829

RSA Data Security, Inc. has provided the following statement with regard to this patent:

> It is our understanding that the proposed PKIX Certificate Profile (PKIX-1) standard currently under review contemplates the use of U.S Patent 4,405,829 entitled "Cryptographic Communication System and Method" (the "RSA patent") which patent is controlled by RSA.
>
> It is RSA's business practice to make licenses to its patents available on reasonable and nondiscriminatory terms. Accordingly, if the foregoing identified IETF standard is adopted, RSA is willing, upon request, to grant non-exclusive licenses to such patent on reasonable and non-discriminatory terms and conditions to those who respect RSA's intellectual property rights and subject to RSA's then current royalty rate

for the patent licensed. The royalty rate for the RSA patent is presently set at 2% of the licensee's selling price for each product covered by the patent. Any requests for license information may be directed to:

Director of Licensing RSA Data Security, Inc. 100 Marine Parkway, Suite 500 Redwood City, CA 94065

A license under RSA's patent(s) does not include any rights to know-how or other technical information or license under other intellectual property rights. Such license does not extend to any activities which constitute infringement or inducement thereto. A licensee must make his own determination as to whether a license is necessary under patents of others.

Diffie-Hellman Key Agreement and Hellman-Merkle Public Key Cryptography

Patent No. 4,200,770: Cryptographic Apparatus and Method ("Diffie-Hellman") expired on August 19, 1997. Patent No. 4,218,582: Public Key Cryptographic Apparatus and Method ("Hellman-Merkle") expired on April 29, 1997.

Appendix A. ASN.1 Structures and OIDs

```
PKIX1 DEFINITIONS IMPLICIT TAGS::=

BEGIN

-- UNIVERSAL Types defined in '93 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 10646-1
--
-- Proposed PKIX 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 }

-- pkix private extensions
id-pe-authorityInfoAccess OBJECT IDENTIFIER ::= { id-pe 1 }

-- policyQualifierIds for Internet policy qualifiers
id-qt-cps      OBJECT IDENTIFIER ::=  { id-qt 1 }
id-qt-unotice  OBJECT IDENTIFIER ::=  { id-qt 2 }

-- 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 }

-- access descriptors for authority info access extension
id-ad-ocsp      OBJECT IDENTIFIER ::= { id-ad 1 }
id-ad-caIssuers OBJECT IDENTIFIER ::= { id-ad 2 }

-- attribute data types --

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

AttributeType           ::=   OBJECT IDENTIFIER

AttributeValue          ::=   ANY

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

-- 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..maxSize)),
        printableString         PrintableString (SIZE (1..maxSize)),
        universalString         UniversalString (SIZE (1..maxSize)),
      bmpString               BMPString (SIZE(1..maxSIZE))
                            }

-- certificate and CRL specific structures begin here

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

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

-- Extension ::= { {id-ce 15}, ... , keyUsage }

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

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 ::= KeyIdentifier

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

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

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

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 }

PolicyQualifierId ::= OBJECT IDENTIFIER

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

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

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

SubjectAltName ::= GeneralNames

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

GeneralName ::= CHOICE {
-- OTHER-NAME ::= TYPE-IDENTIFIER  note: not supported in '88 ASN.1
     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 ::= SEQUENCE {
```

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

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

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

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

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

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)

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

PolicyConstraints ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
     requireExplicitPolicy           [0] SkipCerts OPTIONAL,
     inhibitPolicyMapping            [1] SkipCerts OPTIONAL }

SkipCerts ::= INTEGER (0..MAX)

-- cRLDistributionPoints CRLDistPointsSyntax ::=
--              SEQUENCE SIZE (1..MAX) OF DistributionPoint
```

```
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) }

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

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

KeyPurposeId ::= OBJECT IDENTIFIER

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

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

-- CRL structures

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

TBSCertList  ::=  SEQUENCE  {
     version                 Version OPTIONAL,
                                  -- if present, must be v2
     signature               AlgorithmIdentifier,
     issuer                  Name,
     thisUpdate              Time,
     nextUpdate              Time  OPTIONAL,
     revokedCertificates     SEQUENCE OF SEQUENCE  {
```

```
          userCertificate         CertificateSerialNumber,
          revocationDate          Time,
          crlEntryExtensions      Extensions OPTIONAL
                                         -- if present, must be v2
                               }  OPTIONAL,
     crlExtensions           [0]  EXPLICIT Extensions OPTIONAL
                                         -- if present, must be v2
                               }

-- 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


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

CRLNumber ::= INTEGER (0..MAX)

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

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

BaseCRLNumber ::= CRLNumber

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) }

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

CertificateIssuer ::= GeneralNames

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

HoldInstructionCode ::= OBJECT IDENTIFIER

-- ANSI x9 arc holdinstruction arc

member-body ID ::= { iso 2 }
us ID ::= { member-body 840 }
x9cm ID ::= { us 10040 }
holdInstruction ID ::= {x9cm 2}

-- ANSI X9 holdinstructions referenced by this standard

id-holdinstruction-none ID ::= {holdInstruction 1}
id-holdinstruction-callissuer ID ::= {holdInstruction 2}
id-holdinstruction-reject ID ::= {holdInstruction 3}

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

InvalidityDate ::=  GeneralizedTime

-- Algorithm structures

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

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

     sha1WithRSASignature OBJECT IDENTIFIER  ::=  {
         iso(1) identified-organization(3) oiw(14) secsig(3)
         algorithm(2) 29  }

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

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

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

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

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

     DHParameter ::= SEQUENCE {
          prime INTEGER, -- p
          base INTEGER -- g
                }

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

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

     id-keyEncryptionAlgorithm  OBJECT IDENTIFIER   ::=
          { 2 16 840 1 101 2 1 1 22 }

     KEA-Parms-Id     ::= OCTET STRING

id-ce-subjectKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 14 }
id-ce-keyUsage OBJECT IDENTIFIER ::=  { id-ce 15 }
id-ce-authorityKeyIdentifier OBJECT IDENTIFIER ::=  { id-ce 35 }

CPSuri ::= IA5String

UserNotice ::= CHOICE {
  visibleString     VisibleString,
  bmpString         BMPString
                      }

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}

-- x400 address syntax starts here
```

```
--      OR Names

ORAddressAndOrDirectoryName ::= ORName

ORAddressAndOptionalDirectoryName ::= ORName

ORName ::= [APPLICATION 0] SEQUENCE {
   -- address -- COMPONENTS OF ORAddress,
   directory-name [0] Name OPTIONAL }

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 ::= EXTENSION-ATTRIBUTE
```

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

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 }

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
--  must be regarded as mandatory
```

```
--  from Annex B of ITU-T X.411
--  Reference Definition of MTS Parameter Upper Bounds

--      Upper Bounds
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
```

Appendix B. 1993 ASN.1 Structures and OIDs

```
PKIX1 DEFINITIONS IMPLICIT TAGS::=

BEGIN

--
-- Proposed PKIX 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 }

-- pkix private extensions
id-pe-authorityInfoAccess OBJECT IDENTIFIER ::= { id-pe 1 }

-- policyQualifierIds for Internet policy qualifiers
id-qt-cps      OBJECT IDENTIFIER ::=  { id-qt 1 }
id-qt-unotice  OBJECT IDENTIFIER ::=  { id-qt 2 }

-- 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 }

-- access descriptors for authority info access extension
id-ad-ocsp      OBJECT IDENTIFIER ::= { id-ad 1 }
id-ad-caIssuers OBJECT IDENTIFIER ::= { id-ad 2 }

-- attribute data types --

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

AttributeType           ::=     OBJECT IDENTIFIER

AttributeValue          ::=     ANY

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

```
AttributeValueAssertion ::=     SEQUENCE {AttributeType, AttributeValue}

-- 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..maxSize)),
        printableString         PrintableString (SIZE (1..maxSize)),
        universalString         UniversalString (SIZE (1..maxSize)),
      bmpString               BMPString (SIZE(1..maxSIZE))
                            }

-- from AuthenticationFramework
--    {joint-iso-ccitt ds(5) modules(1) authenticationFramework(7) 2}
-- note this module was defined with EXPLICIT TAGS

-- types --

Certificate            ::=    EXPLICIT 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 must be  v1 or v2--
subjectUniqueIdentifier [2]   IMPLICIT UniqueIdentifier OPTIONAL,
                              ---if present, version must be v1 or v2--
extensions              [3]   Extensions Optional
                              --if present, version must be v3--}  }


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

CertificateSerialNumber ::=     INTEGER
```

```
Algorithmidentifier     ::=     SEQUENCE{
algorithm               ALGORITHM.&id({SupportedAlgorithms}),
parameters              ALGORITHM.&Type({SupportedAlgorithms}
                                             { @algorithm}) OPTIONAL }

--      Definition of the following information object is deferred.
--      SupportedAlgorithms     ALGORITHM  ::=  { ...|... }

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 --

-- Definition of the following information object set is deferred,
-- The set is required to specify a table constraint on the critical
-- component of Extension.
--      ExtensionSet    EXTENSION       ::=     { ... | ... }

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


CertificateList ::=    EXPLICIT SIGNED { SEQUENCE {
```

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


-- information object classes --

ALGORITHM       ::=     TYPE-IDENTIFIER

-- Parameterized Types --
HASHED {ToBeHashed}     ::=     OCTET STRING ( CONSTRAINED-BY {
    --must be the result of applying a hashing procedure to the --
    --DER-encoded octets of a value of -- ToBeHashed })

ENCRYPTED { ToBeEnciphered}     :=      BIT STRING ( CONSTRAINED BY {
    --must be the result of applying an encipherment procedure to the --
    --BER-encoded octets of a value of -- ToBeEnciphered })

SIGNED { ToBeSigned }   ::=     SEQUENCE{
        ToBeSigned,
        COMPONENTS OF SIGNATURE { ToBeSigned }),

SIGNATURE { OfSignature }       ::=     SEQUENCE {
        AlgorithmIdentifier,
        ENCRYPTED { HASHED { OfSignature }}}

-- Key and policy information extensions --

authorityKeyIdentifier EXTENSION ::= {
        SYNTAX          AuthorityKeyIdentifier
        IDENTIFIED BY   { id-ce 35 } }

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 14 } }

SubjectKeyIdentifier ::= KeyIdentifier

keyUsage EXTENSION ::= {
        SYNTAX  KeyUsage
        IDENTIFIED BY { id-ce 15 } }

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

privateKeyUsagePeriod EXTENSION ::= {
        SYNTAX  PrivateKeyUsagePeriod
        IDENTIFIED BY { id-ce 16 } }

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 32 } }

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 ::= { ... }

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 33 } }

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

supportedAlgorithms ATTRIBUTE ::= {
        WITH SYNTAX SupportedAlgorithm
        EQUALITY MATCHING RULE algorithmIdentifierMatch
        ID { id-at 52 } }

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

-- Certificate subject and certificate issuer attributes extensions --

subjectAltName EXTENSION ::= {
        SYNTAX  GeneralNames
        IDENTIFIED BY { id-ce 17 } }

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 18 } }

subjectDirectoryAttributes EXTENSION ::= {
        SYNTAX  AttributesSyntax
        IDENTIFIED BY { id-ce 9 } }

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


-- Certification path constraints extensions --

basicConstraints EXTENSION ::= {
        SYNTAX  BasicConstraintsSyntax
        IDENTIFIED BY { id-ce 19 } }

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

nameConstraints EXTENSION ::= {
        SYNTAX  NameConstraintsSyntax
        IDENTIFIED BY { id-ce 30 } }

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 36 } }

PolicyConstraints Syntax ::= SEQUENCE SIZE (1..MAX) OF SEQUENCE {
        requireExplicitPolicy   [0] SkipCerts OPTIONAL,
        inhibitPolicyMapping    [1] SkipCerts OPTIONAL }

SkipCerts ::= INTEGER (0..MAX)

-- Basic CRL extensions --

cRLNumber EXTENSION ::= {
        SYNTAX  CRLNumber
        IDENTIFIED BY { id-ce 20 } }

CRLNumber ::= INTEGER (0..MAX)

reasonCode EXTENSION ::= {
        SYNTAX  CRLReason
        IDENTIFIED BY { id-ce 21 } }

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 23 } }

HoldInstruction ::= OBJECT IDENTIFIER

invalidityDate EXTENSION ::= {
        SYNTAX  GeneralizedTime
        IDENTIFIED BY { id-ce 24 } }


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

cRLDistributionPoints EXTENSION ::= {
        SYNTAX  CRLDistPointsSyntax
        IDENTIFIED BY   { 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) }

issuingDistributionPoint EXTENSION ::= {
        SYNTAX  IssuingDistPointSyntax
        IDENTIFIED BY   { id-ce 28 } }

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 29 } }

deltaCRLIndicator EXTENSION ::= {
        SYNTAX          BaseCRLNumber
        IDENTIFIED BY   { id-ce 27 } }

BaseCRLNumber ::= CRLNumber

deltaRevocationList ATTRIBUTE ::= {
        WITH SYNTAX     CertificateList
        EQUALITY MATCHING RULE certificateListExactMatch
        ID      {id-at 53 } }


-- Object identifier assignments --

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

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

-- PKIX 1 extensions

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

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

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

CPSuri ::= IA5String

UserNotice ::= CHOICE {
  visibleString     VisibleString,
  bmpString         BMPString
                      }

-- misc missing ASN.1

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}


-- The following OBJECT IDENTIFIERS are not used by this specification:
```

```
-- {id-ce 2}, {id-ce 3}, {id-ce 4}, {id-ce 5}, {id-ce 6}, {id-ce 7},
-- {id-ce 8}, {id-ce 10}, {id-ce 11}, {id-ce 12}, {id-ce 13},
-- {id-ce 22}, {id-ce 25}, {id-ce 26}

-- X.400, Algorithm Identifier, and maximum values Module

ORAddressAndOrDirectoryName ::= ORName

ORAddressAndOptionalDirectoryName ::= ORName

ORName ::= [APPLICATION 0] SEQUENCE {
   -- address -- COMPONENTS OF ORAddress,
   directory-name [0] Name OPTIONAL }

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 }

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

--      Upper Bounds
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
```

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.

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 two annotated hex dumps 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. The first hex dump is a basic dump of the ASN.1 encoding and does not not reflect the fact that the object is a certificate. The second dump

identfies the values of the various certificate fields.

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. The first hex dump is a basic dump of the ASN.1 encoding and does not not reflect the fact that the object is a certificate. The second dump identfies the values of the various certificate fields.

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 certifcates includes the end entity certificate presented in D.2. The hex dump is a basic dump of the ASN.1 encoding.

D.1 Certificate

This section contains an annotated hex dump of a 662 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; and
(g) the certificate is a CA certificate (as indicated through the basic constraints extension.)

D.1.1 ASN.1 Dump of "Self-Signed" Certificate

```
get 0, len=662 (662 bytes in file)
0000 30 82 02 92  658: SEQUENCE
0004 30 82 02 52  594: . SEQUENCE
0008 a0 03          3: . . [0]
0010 02 01          1: . . . INTEGER 2
0013 02 01          1: . . INTEGER 17
0016 30 09          9: . . SEQUENCE
0018 06 07          7: . . . OID 1.2.840.10040.4.3: dsa-with-sha
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
```

```
0038 13 02          2: . . . . . . PrintableString  'US'
0042 31 0c         12: . . . . SET
0044 30 0a         10: . . . . . SEQUENCE
0046 06 03          3: . . . . . . OID 2.5.4.10: O
0051 13 03          3: . . . . . . PrintableString  'gov'
0056 31 0d         13: . . . . SET
0058 30 0b         11: . . . . . SEQUENCE
0060 06 03          3: . . . . . . OID 2.5.4.11: OU
0065 13 04          4: . . . . . . PrintableString  'nist'
0071 30 1e         30: . . . SEQUENCE
0073 17 0d         13: . . . . UTCTime  '970630000000Z'
0088 17 0d         13: . . . . UTCTime  '971231000000Z'
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
0114 13 02          2: . . . . . . PrintableString  'US'
0118 31 0c         12: . . . . SET
0120 30 0a         10: . . . . . SEQUENCE
0122 06 03          3: . . . . . . OID 2.5.4.10: O
0127 13 03          3: . . . . . . PrintableString  'gov'
0132 31 0d         13: . . . . SET
0134 30 0b         11: . . . . . SEQUENCE
0136 06 03          3: . . . . . . OID 2.5.4.11: OU
0141 13 04          4: . . . . . . PrintableString  'nist'
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
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 0d          13: . . [3]
0589 30 0b          11: . . . SEQUENCE
0591 30 09           9: . . . . SEQUENCE
0593 06 03           3: . . . . . OID 2.5.29.19: basicConstraints
0598 04 02           2: . . . . . OCTET STRING
                       : 30 00
0602 30 09           9: . SEQUENCE
0604 06 07           7: . . OID 1.2.840.10040.4.3: dsa-with-sha
0613 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

------- extensions ----------

printber -s 456 pkix-ex1.ber
get 0, len=131 (662 bytes in file)
0000 02 81 80     128: INTEGER
                       : 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
```

D.1.2 Pretty Print of "Self-Signed" Certificate

```
----------
decode: 0-OK, len=662 (662 bytes in file)

      Version: v3
Serial Number: 17
```

```
Signature Alg: dsa-with-sha (1.2.840.10040.4.3)
       Issuer: C=US, O=gov, OU=nist
     Validity: from 970630000000Z
                 to 971231000000Z
      Subject: OU=nist, O=gov, C=US
SubjectPKInfo: dsa (1.2.840.10040.4.1)
       params:
        02 81 80 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 02 14 a7 83 9b f3 bd 2c 20 07 fc 4c e7
        e8 9f f3 39 83 51 0d dc dd 02 81 80 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
   Public Key:
        00 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
    issuerUID:
   subjectUID:
 1 extensions:
     Exten  1:   basicConstraints (2.5.29.19)
        30 00
Signature Alg: dsa-with-sha (1.2.840.10040.4.3)
    Sig Value: 368 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
```

------- extensions ----------

```
printber -s 616 pkix-ex1.ber
get 0, len=46 (662 bytes in file)
0000 30 2c         44: SEQUENCE
0002 02 14         20: . INTEGER
                     : 9d 2d 0c 75 ec ce 01 79 25 4c cd 7b dc fc 17 0e
                     : 0f 2a 22 ef
0024 02 14         20: . INTEGER
                     : 80 61 6f fb dc 71 cf 3f 09 62 b4 aa ad 4b 8c 28
                     : 68 d7 60 fe
```

D.2 Certificate

This section contains an annotated hex dump of a xxx 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 and will expire on December 1, 1997;
(f) the certificate contains a 1024 bit DSA public key;
(g) the certificate is an end entity certificate unless external information is provided, as the basic constraints extension is not present;
(h) the certificate includes one alternative name - an RFC 822 address.

D.2.1 Basic ASN.1 Dump of "End Entity" Certificate

```
----------

get 0, len=697 (697 bytes in file)
0000 30 82 02 b5  693: SEQUENCE
0004 30 82 02 75  629: . SEQUENCE
0008 a0 03          3: . . [0]
0010 02 01          1: . . . INTEGER 2
0013 02 01          1: . . INTEGER 18
0016 30 09          9: . . SEQUENCE
0018 06 07          7: . . . OID 1.2.840.10040.4.3: dsa-with-sha
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
0038 13 02          2: . . . . . PrintableString  'US'
```

```
0042 31 0c           12: . . . SET
0044 30 0a           10: . . . . SEQUENCE
0046 06 03            3: . . . . . OID 2.5.4.10: O
0051 13 03            3: . . . . . PrintableString  'gov'
0056 31 0d           13: . . . SET
0058 30 0b           11: . . . . SEQUENCE
0060 06 03            3: . . . . . OID 2.5.4.11: OU
0065 13 04            4: . . . . . PrintableString  'nist'
0071 30 1e           30: . . SEQUENCE
0073 17 0d           13: . . . UTCTime  '970730000000Z'
0088 17 0d           13: . . . UTCTime  '971201000000Z'
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
0114 13 02            2: . . . . . PrintableString  'US'
0118 31 0c           12: . . . SET
0120 30 0a           10: . . . . SEQUENCE
0122 06 03            3: . . . . . OID 2.5.4.10: O
0127 13 03            3: . . . . . PrintableString  'gov'
0132 31 0d           13: . . . SET
0134 30 0b           11: . . . . SEQUENCE
0136 06 03            3: . . . . . OID 2.5.4.11: OU
0141 13 04            4: . . . . . PrintableString  'nist'
0147 31 11           17: . . . SET
0149 30 0f           15: . . . . SEQUENCE
0151 06 03            3: . . . . . OID 2.5.4.3: CN
0156 13 08            8: . . . . . PrintableString  'Tim Polk'
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
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 1d            29: . . [3]
0608 30 1b            27: . . . SEQUENCE
0610 30 19            25: . . . . SEQUENCE
0612 06 03             3: . . . . . OID 2.5.29.17: subjectAltName
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 09             9: . SEQUENCE
0639 06 07             7: . . OID 1.2.840.10040.4.3: dsa-with-sha
0648 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

-------- extensions ----------

printber -s 475 pkix-ex2.ber
get 0, len=131 (697 bytes in file)
0000 02 81 80        128: INTEGER
                          : 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
```

D.2.2 Pretty Print of "End Entity" Certificate

```
----------
decode: 0-OK, len=697 (697 bytes in file)
```

```
      Version: v3
Serial Number: 18
Signature Alg: dsa-with-sha (1.2.840.10040.4.3)
       Issuer: C=US, O=gov, OU=nist
     Validity: from 970730000000Z
                 to 971201000000Z
      Subject: CN=Tim Polk, OU=nist, O=gov, C=US
SubjectPKInfo: dsa (1.2.840.10040.4.1)
       params:
        02 81 80 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 02 14 a7 83 9b f3 bd 2c 20 07 fc 4c e7
        e8 9f f3 39 83 51 0d dc dd 02 81 80 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
   Public Key:
        00 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
    issuerUID:
   subjectUID:
 1 extensions:
     Exten  1:   subjectAltName (2.5.29.17)
        30 10 81 0e 77 70 6f 6c 6b 40 6e 69 73 74 2e 67
        6f 76
Signature Alg: dsa-with-sha (1.2.840.10040.4.3)
    Sig Value: 368 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
```

```
-------- extensions ----------

printber -s 619 pkix-ex2.ber
get 0, len=18 (697 bytes in file)
0000 30 10          16: SEQUENCE
0002 81 0e          14: . [1]
                      : 77 70 6f 6c 6b 40 6e 69 73 74 2e 67 6f 76
Note: This subjectAltName data is IMPLICIT TAGS - is that correct?

printber -s 651 pkix-ex2.ber
get 0, len=46 (697 bytes in file)
0000 30 2c          44: SEQUENCE
0002 02 14          20: . INTEGER
                      : 2b 82 c9 2d 79 9c a4 16 97 22 b1 48 16 03 c2 ed
                      : 31 65 99 d5
0024 02 14          20: . INTEGER
                      : 3f 90 79 17 f8 9d 50 fb f3 5d 70 b7 40 31 a3 74
                      : 31 d7 b1 30
```

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 2;
(b) the certificate is signed with RSA and the MD5 hash algorithm;
(c) the issuer's distinguished name is OU=esCert-UPC;O=UPC;L=Barcelona;STREET=Catalunya;C=ES
(d) and the subject's distinguished name is CN=escert.upc.es;OU=esCert-UPC;O=UPC;L=Barcelona;STREET=Catalunya;C=ES
(e) the certificate was issued on May 21, 1996 and will expire on May 21, 1997;
(f) the certificate contains a 768 bit RSA public key which is intended for generation of digital signatures;
(g) the certificate is an end entity certificate (not a CA certificate);
(h) the certificate includes two alternative names - an RFC 822 address, and a URL.

```
 sequence length 029f=671 bytes
 30 82 02 9f
    sequence length 0208h=520 bytes
    30 82 02 08
       explicit tag 00 "Version"
       a0 03
          integer length 1 value 2 [version is 3]
          02 01 02
       integer length 1 value 2 [serial number 2]
```

```
       02 01 02
       sequence length 13 [signature]
       30 0d
          object identifier length 9 {1 2 840 113549 1 1 4}
                                {iso(1) member-body(2) us(840) etc.}
          06 09 2a 86 48 86 f7 0d 01 01 04
          null [null parameters]
          05 00
       sequence length 88 [issuer]
          30 58
             RDN length 11
             31 0b
                sequence length 9
                30 09
                   object identifier length 3  { 2 5 4 6 }
                   06 03 55 04 06
                   printable string length 2 "ES"
                   13 02 45 53
             RDN length 18
             31 12
                sequence length 16
                30 10
                   object identifier length 3 { 2 5 4 9 }
                   06 03 55 04 09
                   printable string length 9 "Catalunya"
                   13 09 43 61 74 61 6c 75 6e 79 61
             RDN length 18
             31 12
                sequence length 16
                30 10
                   object identifier length 3 { 2 5 4 7 }
                   06 03 55 04 07
                   printable string length 9 "Barcelona"
                   13 09 42 61 72 63 65 6c 6f 6e 61
             RDN length 12
             31 0c
                sequence length 10
                30 0a
                   object identifier {2 5 4 10 }
                   06 03 55 04 0a
                   printable string length 3 "UPC"
                   13 03 55 50 43
             RDN length 19
             31 13
                sequence length 17
                30 11
                   object identifier {2 5 4 13 }
                   06 03 55 04 0b
```

```
                   printable string length 10 "esCERT-UPC"
                   13 0a 65 73 43 45 52 54 2d 55 50 43
       sequence length 0x1e= 30
          30 1e
             UTCTime "960521095826Z"
             17 0d 39 36 30 35 32 31 30 39 35 38 32 36 5a
             UTCTime "979521095826Z"
             17 0d 39 37 30 35 32 31 30 39 35 38 32 36 5a
       sequence length
       30 70
          31 0b
             30 09
                { 2 5 4 6 }
                06 03 55 04 06
                "ES"
                13 02 45 53
          RDN
          31 12
             30 10
                { 2 5 4 9 }
                06 03 55 04 09
                "Catalunya"
                13 09 43 61 74 61 6c 75 6e 7961
          RDN
          31 12
             30 10
                { 2 5 4 7 }
                06 03 55 04 07
                "Barcelona"
                13 09 42 61 72 63 65 6c 6f 6e 61
          RDN
          31 0c
             30 0a
                { 2 5 4 10 }
                06 03 55 04 0a
                "UPC"
                13 03 55 50 43
          RDN
          31 13
             30 11
                { 2 5 4 11 }
                06 03 55 04 0b
                "esCERT-UPC"
                13 0a 65 73 43 45 52 54 2d 55 50 43
          RDN
          31 16
             30 14
                { 2 5 4 3 }
```

```
                06 03 55 04 03
                "escert.upc.es"
                13 0d 65 73 63 65 72 74 2e 75 70 63 2e 65 73
        subjectPublicKeyInfo
          30 7c
             algorithmIdentifier
             30 0d
                { 1 2 840 113549 1 1 1}
                06 09 2a 86 48 86 f7 0d 01 01 01
                null parameters
                05 00
             { subject's public key }
             03 6b  BIT STRING length 107 bytes (856 bits)
                               0030 6802 6100 beaa 8b77 54a3 afca 779f
                               2fb0 cf43 88ff a66d 7955 5b61 8c68 ec48
                               1e8a 8638 a4fe 19b8 6217 1d9d 0f47 2cff
                               638f 2991 04d1 52bc 7f67 b6b2 8f74 55c1
                               3321 6c8f ab01 9524 c8b2 7393 9d22 6150
                               a935 fb9d 5750 32ef 5652 5093 abb1 8894
                               7856 15c6 1c8b 0203 0100 01
         explicit tag 3 "extensions" length 0x84=132
         a3 81 84
            sequence 129 bytes
            30 81 81
               sequence 12 bytes
               30 0b
                  id-ce-keyUsage = { 2 5 29 15 }
                  06 03 55 1d 0f
                  by default, critical = FALSE
                  octet string
                  04 04 03 02 07 80
               30 09
                  id-ce-basicConstraints = { 2 5 29 19 }
                  06 03 55 1d 13
                  by default, critical = FALSE
                  octet string
                  04 02
                     null sequence - by default, subject is end entity
                     30 00
               30 3d
                  id-ce-subjectAltName = { 2 5 29 17 }
                  06 03 55 1d 11
                  by default, critical = FALSE
                  octet string
                  04 36
                     30 34
                        rfc822name
                        a1 1a
```

```
                        IA5String "escert-upc@escert.upc.es"
                        16 18 65 73 63 65 72 74 2d 75 70 63 40 65 73 63
                        65 72 74 2e 75 70 63 2e 65 73
                     uniformResourceIdentifier
                     a6 16
                        IA5String "http://escert.upc.es"
                        16 14 68 74 74 70 3a 2f 2f 65 73 63 65 72 74 2e
                        75 70 63 2e 65 73
            30 28
               id-ce-certificatePolicies = { 2 5 29 32 }
               06 03 55 1d 20
               by default, critical = FALSE
               octet string
               04 21
                  30 1f
                     30 1d
                        06 04 2a 84 80 00
                        { 2 2 32768 }
                  30 15
                     30 07
                        { 2 2 32768 1 }
                        06 05 2a 84 80 00 01
                      30 0a
                         { 2 2 32768 2 }
                         06 05 2a 84 80 00 02
                         02 01 0a
   sequence
   30 0d
      { 1 2 840 113549 1 1 4 }
      06 09 2a 86 48 86 f7 0d 01 01 04
      null parameters
      05 00
   bit string length 129  (signature)
   03 81 81 005b fdc2 a704 d483 4e17 6da6 fa27 e7c6
            f8ab b95d 9fd0 a1df d797 9fe0 20a6 c57a
            64cd 522f e9ae dabe 9ce4 d597 edf1 84c0
            d0fe 9bef 54b1 80e5 bf3c c9ed 9320 2d52
            21e9 bcb9 e34f ac11 650e 8fa1 6899 6347
            e53d e442 7313 fac5 c834 8cc0 4118 89d5
            e6a0 185b 5d86 1c1e c670 d80e 8964 9483
            8e3b 407c 59cf 2b2f b7ce 9798 1215 ef13
            d4
```

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.

```
printber pkix-crl.ber
get 0, len=189 (189 bytes in file)
0000 30 81 ba     186: SEQUENCE
0003 30 7c        124: . SEQUENCE
0005 02 01          1: . . INTEGER 1
0008 30 09          9: . . SEQUENCE
0010 06 07          7: . . . OID 1.2.840.10040.4.3: dsa-with-sha
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
0030 13 02          2: . . . . . PrintableString  'US'
0034 31 0c         12: . . . SET
0036 30 0a         10: . . . . SEQUENCE
0038 06 03          3: . . . . . OID 2.5.4.10: O
0043 13 03          3: . . . . . PrintableString  'gov'
0048 31 0d         13: . . . SET
0050 30 0b         11: . . . . SEQUENCE
0052 06 03          3: . . . . . OID 2.5.4.11: OU
0057 13 04          4: . . . . . PrintableString  'nist'
0063 17 0d         13: . . UTCTime  '970801000000Z'
0078 17 0d         13: . . UTCTime  '970808000000Z'
0093 30 22         34: . . SEQUENCE
0095 30 20         32: . . . SEQUENCE
0097 02 01          1: . . . . INTEGER 18
0100 17 0d         13: . . . . UTCTime  '970731000000Z'
0115 30 0c         12: . . . . SEQUENCE
0117 30 0a         10: . . . . . SEQUENCE
0119 06 03          3: . . . . . . OID 2.5.29.21: reasonCode
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
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


printber -s 143 pkix-crl.ber
get 0, len=46 (189 bytes in file)
0000 30 2c         44: SEQUENCE
0002 02 14         20: . INTEGER
                     : 9e d8 6b c1 7d c2 c4 02 f5 17 84 f9 9f 46 7a ca
```

```
                         : cf b7 05 8a
0024 02 14            20: . INTEGER
                         : 9e 43 39 85 dc ea 14 13 72 93 54 5d 44 44 e5 05
                         : fe 73 9a b2
```

Security Considerations

This entire memo is about security mechanisms.

Author Addresses:

Russell Housley
SPYRUS
PO Box 1198
Herndon, VA 20172
USA
housley@spyrus.com

Warwick Ford
VeriSign, Inc.
One Alewife Center
Cambridge, MA 02140
USA
wford@verisign.com

Tim Polk
NIST
Building 820, Room 426
Gaithersburg, MD 20899
USA
wpolk@nist.gov

David Solo
BBN
150 CambridgePark Drive
Cambridge, MA 02140
USA
solo@bbn.com