TWG-98-07

Federal Public Key Infrastructure (PKI)

X.509 Certificate and CRL Extensions Profile

March 9, 1998

Prepared By :

BOOZ•ALLEN & HAMILTON INC.
900 Elkridge Landing Road
Linthicum, Maryland 21090

Table of Contents

Page No.

0 Introduction

This document specifies the Federal Public Key Infrastructure (FPKI) Version 3 (V3) X.509 certificates and Version 2 (V2) Certificate Revocation Lists (CRL) as described in [1]. Implementation guidance is provided for certificate generation entities (e.g., Certification Authority [CA]) and certificate processing entities (e.g., User Agent [UA]). Throughout this document the term “CA” will represent any implementation that can create certificates; the term “authorizations” will be defined as the clearances and privileges asserted in user certificates by a CA.

V3 X.509 certificates contain the identity and attribute data of a subject using the base certificate with applicable extensions. The base certificate contains such information as the version number of the certificate, the certificate’s identifying serial number, the signature algorithm used to sign the certificate, the issuer’s distinguished name, the validity period of the certificate, the distinguished name of the subject, and information about the subject’s public key. To this base certificate is appended numerous certificate extensions. This document describes and stipulates those extensions which are required for FPKI-compliant systems. More detailed information about X.509 certificates can be found in Recommendation X.509.

0.1 Structure

The document is divided into three sections. Sections 1 and 2 describe the V3 certificates and the V2 CRLs, respectively. These sections describe the extensions implemented by FPKI including the values contained in FPKI certificates, CRLs, and extensions, as well as the recommended interpretation of the extensions. Guidance is provided to indicate those extensions that may be added to FPKI certificates and CRLs. This guidance is not intended to require the extensions be present in every certificate (except as specifically indicated). In addition, Section 2 also describes the validation process for CRLs and Indirect CRLs (ICRLs). Section 3 specifies the FPKI profile for the V3 X.509 certificate and V2 CRLs. This profile lists the protocol elements FPKI CAs must generate to add extensions to a certificate and the protocol elements FPKI certificate processing entities must understand if the extension is to be processed properly. Finally, Appendices A and B discuss the processing of algorithm identifiers and the parameters used within the Digital Signature Algorithm (DSA), and the Key Exchange Algorithm (KEA), respectively.

0.2 Acronyms

ASN.1	Abstract Syntax Notation 1
CA	Certification Authority
CRL	Certificate Revocation List
DN	Distinguished Name
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
EE	End Entity
FPKI	Federal Public Key Infrastructure
ICRL	Indirect Certificate Revocation List
ITU-T	International Telecommunications Union Telecommunications Sector
KEA	Key Exchange Algorithm
OID	Object Identifier
PRBAC	Partition Rule Based Access Control
RFC	Request For Comments
UA	User Agent
V2	Version 2
V3	Version 3

0.3 References

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

[2] SDN.702: *Abstract Syntax for Utilization with Common Security Protocol (CSP), Version 3 X.509 Certificates and Version 2 Certificate Revocation Lists, Revision 3 of Prototype Baseline,* 31 July 1997.

[3] SDN.706, *X.509 Certificate and Certificate Revocation List Profiles and Certificate Path Processing Rules for the Multilevel Information Systems Security Initiative,* Revision B, 24 October 1997.

[4] SDN.801, *Access Control Concepts and Mechanisms,* Revision B, 24 October 1997.

[5] ITU-T Recommendation X.690, *Information Technology - ASN.1 Encoding Rules - Specification of Basic Encoding Rules, Canonical Encoding Rules and Distinguished Encoding Rules.*

[6] *Public Key Infrastructure (PKI) Technical Specifications (Version 2.2): Part C - Concept of Operations*, TWG-xx, 1998.

[7] Internet Public Key Infrastructure, *Part I: X.509 Certificate and CRL Profile*, Internet Draft.

1 V3 Certificates

CAs create certificates for user authentication procedures that require one user to obtain another user's public key. So that users trust the public key, the CA employs a digital signature to cryptographically sign certificates and provide assurance that the information within the certificate is correct. The fields in a certificate identify the issuer (i.e., CA), subject (i.e., user), version number, subject's public key, validity period, and serial number of the certificate along with the public key algorithm used to certify the certificate. A CA may also add certificate extensions containing additional information about the user or the CA (see section 1.2) depending on the implementation.

This document stipulates the required certificate and CRL format for FPKI-compliant programs. Any specific program implementing certificate-based public key cryptography, and claiming compliance to the Federal Public Key Infrastructure requirements is required to tailor its X.509 certificates (as defined in [1]) within the parameters outlined within this document.

Through the remainder of this document, requirements for generation and processing of particular extensions are applied. The application of requirements is segmented into three types of certificates: a self-signed "root CA" certificate, the CA (which will include Subordinate CAs [SCAs]) certificate, and the End Entity (EE) certificate.

1.1 Base X.509 Certificate Processing

FPKI CAs shall, for all certificates:

- include the version field with an integer value of 2[1] to indicate that the certificate is a version 3 certificate;
- include the serialNumber field with an integer value to indicate the certificate's serial number;
- include in the signature field the identifier (OID) of the algorithm used to sign the certificate, but not populate the parameters in this field;
- include the issuer field with the X.500 distinguished name of the CA who created the certificate;
- include the validity field with the time period for which the certificate is considered valid[2];
- include the subject field with the X.500 distinguished name of the subject to whom the certificate was issued;
- if the subject's DSA parameters are different from the certificate issuer's DSA parameters then they are included in the SubjectPublicKeyInfo algorithm parameters field. For DSA

[1] A value 0 in the version field indicates a version 1 certificate and a value of 1 indicates a version 2 certificate.
[2] Certificate validity dates through the year 2049 shall be encoded as UTCTime; certificate validity dates in 2050 or later shall be encoded as GeneralizedTime.

X.509 certificates, the DSA public key will be ASN.1 encoded as an INTEGER which is then encapsulated with the SubjectPublicKeyInfo subjectKey BIT STRING;

- omit the issuerUniqueIdentifier and subjectUniqueIdentifier fields;
- use the Parametrized Type as defined in X.500 to sign the certificate (the DSA parameters shall not be included in the SIGNED MACRO algorithm identifier field); and
- include the extensions field and extensions as described in section 1.2. The following definition for Extensions is described in the Technical Corrigendum 2 to Recommendation X.509:

```
Extensions  ::=  SEQUENCE of Extension

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

-- Reference [1] specifies a list of extensions that may be included in V3 X.509 certificates and
-- CRLs. ExtensionSet includes the object identifiers for the valid extensions. The set is
required
-- to specify a table constraint on the critical component of Extension.

ExtensionSet   EXTENSION  ::=  { 2 5 29 35 | 2 5 29 14 | 2 5 29 15 | 2 5 29 16 |
                         2 5 29 32 | 2 5 29 33 | 2 5 29 17 | 2 5 29 18 |
                         2 5 29 9 | 2 5 29 19 | 2 5 29 30 | 2 5 29 34 |
                         2 5 29 31 | 2 5 29 20 | 2 5 29 21 | 2 5 29 23 |
                         2 5 29 24 | 2 5 29 28 | 2 5 29 29}

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

Certificate path processing begins with a trusted public key and associated parameters which are obtained in a trusted manner. The trusted key must be maintained in a manner to insure it integrity.

For each certificate in the path, certificate processing entities shall:

- attempt to validate the signature of the certificate;
- process fields generated by a FPKI CA as identified in the certificate profiles;
- ignore the issuerUniqueIdentifier and subjectUniqueIdentifier fields if they are present; and
- process the extension fields, if present, as described within Sections 1.2 and 1.4 of this document.

The Distinguished Encoding Rules (DER), [5], allow several methods for formatting UTCTime and GeneralizedTime. It is important that all implementations use the same format to minimize signature verification problems. To ensure that UTCTimes are consistently formatted, FPKI-compliant software must format all UTCTimes included in ASN.1 syntax's that are encoded using the DER using the 'Z' format and must never omit the "seconds" field (even when it is '00') (i.e., the format shall be YYMMDDHHMMSSZ). The system shall interpret the year field, YY, as 19YY when YY is greater than or equal to 50, and 20YY when YY is less than 50. The GeneralizedTime type for this profile shall be expressed using the "Z" format and shall include seconds (i.e., the format shall be YYMMDDHHMMSSZ) but not fractional seconds.

1.2 Certificate Extensions

V3 certificates provide a mechanism for CAs to append additional information about the subject's public key, issuer's public key, and issuer's CRLs. Standard certificate extensions are defined for V3 X.509 certificates. It is not required that all the extensions be used by FPKI, however all the extensions are included here to insure completeness. Table 3.2.2 provides guidance regarding FPKI extension usage. These extensions provide methods of increasing the amount of information the X.509 certificate conveys to facilitate automated certificate processing. The following sections describe how these extensions are implemented. Note that the Subject Directory Attributes extension (see Section 1.2.10) may be used to implement access control, if desired, based on the Partition Rule Based Access Control (PRBAC) concept defined by MISSI (see [4]).

As described in Section 12.1 of [1] and repeated here, an extension is flagged as being either critical or non-critical. If an extension is flagged critical and a certificate-using system does not recognize the extension field type or does not implement the semantics of the extension, then that system shall consider the certificate invalid. If an extension is flagged non-critical, a certificate-using system that does not recognize or implement that extension type may process the remainder of the certificate ignoring the extension.

In the FPKI all certification paths start from a public key contained in a "root-CA certificate." A root-CA certificate:

- is self-signed, that is, signed with the private key corresponding to the public key contained in the subject public key field of the certificate;
- contains any needed parameters in the subject public key info field, where the digital signature algorithm used in the certificate requires the use of parameters;
- contains few or no extensions;
- is kept in protected memory or otherwise protected from alteration by an intruder;
- is transferred to the application or certificate using system in an authenticated manner. The signature on the root-CA Certificate cannot authenticate the certificate.

A Federal PKI with several administrative hierarchies of CA's is defined in [6]. At the top of each of these hierarchies is an administrative "root CA." These root CAs are cross-certified with each other. Moreover, in the hierarchies, subordinate CAs not only are issued certificates by their superior CA in the hierarchy, they also cross-certify with their superior CA.

A consequence of this PKI topology is that any certificate using system can use a root CA certificate issued by any CA in the FPKI to start it's certification paths, provided the certificate using system trusts that CA and has an authenticated copy of its root CA certificate. That root-CA certificate may be issued by one of the administrative root CAs, described above, but it may also be issued by the "local" CA that issued the certificate using system its own end entity certificate, or by any other CA in the FPKI. Which root-CA certificates may be used by agency certificate using systems to start certification paths is a matter of agency security policy.

Any certificate using system in the FPKI can view any CA in the FPKI as the root CA for starting certification paths, provided:

1. the certificate using system has an authenticated copy of the root-CA certificate, and,
2. local agency security policy allows the use of that root-Ca certificate.

Agencies will designate the CAs whose root-CA certificates may be used by certificate using systems within the agency, and establish the approved mechanisms for obtaining the authenticated root-CA certificates.

Due to patent concerns associated with methods for implementing certificate revocation at the time of this publishing, requirements for and relating to CRLs, indirect CRLs, and CRL partitioning, may be overcome by future circumstance.

1.2.1 authorityKeyIdentifier

This non-critical extension identifies the public key used to verify the signature on a certificate. It enables distinct keys used by the same CA to be differentiated. This extension may hold an explicit key identifier, or an explicit certificate identifier. This extension is useful when a CA uses more than one key (e.g., when the CA key is re-keyed). Use of this extension may provide improved efficiency when attempting to locate a specific certificate. Since this extension is only viewed as an efficiency enhancing extension, it is not required that certificate processing entities be capable of processing this extension. However, use of this extension is highly encouraged, and thus, FPKI CAs shall be capable of populating this extension as described below.

1.2.1.1 Generation Requirements

FPKI CAs shall:

- include the extension in all CA and EE certificates;
- not include the extension in the self-signed certificate;
- omit the authorityCertIssuer and authorityCertSerialNumber fields; and,
- include the authorityKeyIdentifier field with a unique identifier for the CA's material (the subject key identifier of the issuer's certificate).

1.2.1.2 Processing Requirements

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

1.2.2 subjectKeyIdentifier

This non-critical extension identifies the public key being certified. It enables distinct keys used by the same subject to be differentiated. This extension may hold the explicit key identifier, and is useful when a subject uses more than one key. This extension is required in the self-signed as a result of the possibility that several Root-CAs will coexist. Similar to the previous extension, this extension is viewed as an efficiency enhancing extension, and thus, certificate processing entities are not required to provide this capability. However, use of this extension is highly encouraged, and thus, it is required the FPKI CAs be capable of populating this extension as described below.

1.2.2.1 Generation Requirements

FPKI CAs shall:

- include this extension in all certificates including the self-signed certificate; and,
- include the subjectKeyIdentifier as a SHA-1 hash of the certificate's public key.

1.2.2.2 Processing Requirements

There are no requirements for FPKI implementations to process this extension, though it is encouraged that certificate processing entities be capable of recognizing the subjectKeyIdentifier field as the identifier of the public key being certified.

1.2.3 keyUsage

This extension indicates the purposes for which the certified public key is used. It may be implemented as either a critical or a non-critical extension. The KeyUsage field includes bit values used for digital signature verification for purposes other than non-repudiation, certificates, or CRLs; digital signature for non-repudiation; enciphering keys or other security information; enciphering user data; a key agreement mechanism; a CA to sign certificates; and a CA to sign CRLs.

1.2.3.1 Generation Requirements

FPKI CAs shall:

- set the criticality flag to "true;"
- include the extension in all CA and EE;

- not include the extension in self-signed certificates; and
- indicate the appropriate key usage according to the following matrix of valid key usage combinations (note: valid combinations appear as columns in the table):

Key Usages	Valid Combinations			
digitalSignature				x*
nonRepudiation				x*
keyEncipherment	x			
dataEncipherment		x		
keyAgreement			x	
keyCertSign				x*
cRLSign				x*
encipherOnly**	-	-	-	-
decipherOnly**	-	-	-	-

* Any subset combination of these key usages is also valid

** There are no requirements to support these key usages.

Furthermore, the CA shall set the key usage bits based on the following:

- The *digitalSignature* bit should be set when the key is for use in ephemeral applications, e.g., for a single session authentication application.

- The *nonRepudiation* bit should be set when the key is used to sign an object which may require the validation of the signature at a future time.

- If the key may be used for both *digitalSignature* and *nonRepudiation* applications, both bits may be set.

- When the public key is used to provide confidentiality using a key agreement mechanism, e.g., KEA, Diffie-Hellman, or a variant thereof, it is recommended that the key usage *keyAgreement* be set; when the public key is used to provide confidentiality using a key encipherment mechanism, e.g., RSA key transport, it is recommended that the key usage *keyEncipherment* be set.

- When the public key may be used to encipher data other than cryptographic keys, the *dataEncipherment* bit should be set.

- When the public key may be used to sign certificates the *keyCertSign* bit should be set; this bit shall only be set in CA certificates.

- When the public key may be used to sign CRLs, the *cRLSign* bit should be set; this bit shall only be set in CA certificates.

- The *encipherOnly* and *decipherOnly* key usages are intended to provide support for key agreement schemes where separate shared secret keys are used in each direction of communication. In such a scheme, a user has more than one set of key pairs and bits 7 (*encipherOnly*) and 8 (*decipherOnly*) are used to distinguish between the two types. The originator of a message would use the recipient's public key certificate with bits 4 (*keyAgreement*) and 7 (*encipherOnly*) to create a key encryption key. The recipient would use the originator's certificate with bits 4 (*keyAgreement*) and 8 (*decipherOnly*) to create the key encryption key. Typically the originator would pass his own certificate with bits 4 and 8 along with the message. FPKI systems are not required to implement a key management scheme where the symmetric keys used in each direction of communication are derived from separate public key pairs. If such a scheme is *not* implemented, then the *encipherOnly* and *decipherOnly* key usages shall be unset (set to "0").

1.2.3.2 Processing Requirements

Before using a public key certificate, an application shall check the criticality of the Key Usage extension. If the extension is "critical" the application shall ensure the following:

- When the public key is used for an ephermeral application such as a single session authentication application, the *digitalSignature* bit is set to "1";

- When the public key is used to validate the signature of a signed object existing for some period of time, the *nonRepudiation* bit is set to "1";

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

- When the public key is used to provide confidentiality using a key encipherment mechanism, e.g., RSA key transport, the *keyEncipherment* bit is set to "1";

- When the public key is used to encipher data other than cryptographic keys, the *dataEncipherment* bit is set to "1";

- When the public key is used to validate the signature on a certificate, the *keyCertSign* bit is set to "1";

- When the public key is used to validate the signature on a CRL, the *cRLSign* bit is set to "1".

1.2.4 extendedKeyUsage

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

1.2.5 privateKeyUsagePeriod

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

1.2.6 certificatePolicies

This extension lists certificate policies that the certificate is expressly recognized as supporting, together with optional qualifier information pertaining to these policies. It may be implemented as either a critical or a non-critical extension. This field is processed during the certification path validation as described in Section 1.4. The certificate policy indicates the procedures under which the certificate was created.

1.2.6.1 Generation Requirements

FPKI CAs that generate this extension shall:

- set the criticality flag ;
- include the extension in all CA and EE certificates;
- include the PolicyInformation field(s) with the applicable policyIdentifier field(s); and
- include the OID(s) for the applicable certificate policy in the policyIdentifier field(s).

1.2.6.2 Processing Requirements

FPKI certificate processing entities shall:

- process the PolicyInformation field(s) as a collection of policyIdentifier field(s);
- process the certificate policy OID(s) in the policyIdentifier field(s); and
- process the policyQualifierIds id-pkix-cps and id-pkix-unotice (see [7]).

When validating a FPKI certification path, FPKI applications must set the initial-explicit-policy indicator to TRUE. Except for the Root-CA certificate, every FPKI X.509 certificate will include the policyConstraints extension with the requireExplicitPolicy field set with SkipCerts set to zero. Therefore, when processing FPKI certification paths, the FPKI certificate processing entity must

reject any certificate that doesn't include one of the acceptable set of policy identifiers (as described in Section 1.4) in the policyIdentifier field (except for the Root-CA certificate).

1.2.7 policyMappings

This non-critical extension allows a certificate issuer to indicate that one or more of that issuer's certificate policies is considered equivalent to another policy used in the subject CA's domain. The assignment of policy mappings is restricted to CA. This is enforced through the policyConstraints extension (Section 1.2.13). This extension will support cross-certification and hold the OIDs of equivalent policies.

1.2.7.1 Generation Requirements

FPKI CAs shall:

- generate this extension for applicable CA digital signature certificates; and
- include a combination of issuerDomainPolicy field(s) and subjectDomainPolicy field(s) with the applicable CertPolicyId field(s).

1.2.7.2 Processing Requirements

FPKI certificate processing entities shall interpret the combination(s) of issuerDomainPolicy OID and subjectDomainPolicy OID as equivalent. The policies may be added to the applicable policies state variable during certificate path validation as described in Section 1.4.

1.2.8 subjectAltName

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

1.2.8.1 Generation Requirements

FPKI CAs shall:

- set the criticality flag to "false;"
- not include this extension in self-signed certificates;
- generate this extension for applicable CA and EE certificates; and
- be capable of populating GeneralName with types dNSName, directoryName, or uniformResourceIdentifier.

1.2.8.2 Processing Requirements

Though the FPKI has no requirements for generating alternative names, the FPKI shall provide processing support for other communities that support alternative names. FPKI certificate processing entities shall:

- apply the Name Constraints (as described in the Name Constraints and Certification Path Processing Procedure sections) to this extension as part of the certification path validation process;
- if this extension is flagged critical, then reject the certificate that does not include a directoryName in the GeneralName SEQUENCE; and
- if this extension is flagged non-critical, then the GeneralNames SEQUENCE does not need to include a directoryName.

1.2.9 issuerAltName

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

1.2.9.1 Generation Requirements

FPKI CAs shall:

- set the criticality flag to “false;”
- not include this extension in self-signed certificates;
- generate this extension for applicable CA, EE, and Cross certificates; and
- be capable of populating GeneralName with types dNSName or uniformResourceIdentifier.

1.2.9.2 Processing Requirements

Though the FPKI has no requirements for generating alternative names, the FPKI shall provide processing support for other communities that support alternative names. FPKI certificate processing entities shall:

- if this extension is flagged critical, then FPKI certificate validation software must reject a certificate that does not include a directoryName in the GeneralName SEQUENCE; and
- if this extension is flagged non-critical, then the General Names does not need to include a directoryName.

1.2.10 subjectDirectoryAttributes

This non-critical extension may convey any desired directory attribute values for the subject of the certificate.

A syntax has been defined by the Multilevel Information System Security Initiative (MISSI) to use this extension to carry access control related information. It is recommended that Federal CAs desiring to convey clearances, citizenship or other access control information, use the syntax defined by MISSI. CAs not needing to convey authorizations in X.509 certificates need not populate the subjectDirectoryAttributes field.

The type of access control mechanism implemented by MISSI is referred to as Partition Rule-Based Access Control (PRBAC). The latest revisions of [2], [3], and [4] should be referenced for the implementation of PRBAC.

No other attributes have been defined for this extension.

1.2.11 basicConstraints

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

This extension is required in all signature certificates. Since processing of this extension in a "trusted" certificate is at the discretion of the implementor, the implementor may choose to create applications to require every certificate in a certification path, including the "trusted" certificate, to assert that they are a CA within a basic constraints extension. For this reason, this extension shall be included in all self-signed and CA certificates Since the processing of this extension in the self-signed certificate is not considered critical and it is not the intent to have certificate processors reject this certificate for not processing the "trusted" certificate, it is indicated as "non-critical" within the self-signed certificate.

1.2.11.1 Generation Requirements

FPKI CAs shall:

- include this extension in all self-signed, CA, and EE digital signature certificates;
- set the criticality flag to "true" in CA and EE certificates;
- set the criticality flag to "false" in self-signed certificates;
- set the CA Boolean flag to "True" for self-signed and CA certificates, and enter a path length constraint if applicable; and
- use the default CA value (False) for EE certificates.

1.2.11.2 Processing Requirements

FPKI certificate processing entities shall:

- reject processing of certificates issued by entities without the cA Boolean flag set to "True" (except for self-signed certificates);
- impose the pathLenConstraint field, if present; and
- treat the certificate as an EE if this extension is not present.

1.2.12 nameConstraints

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

1.2.12.1 Generation Requirements

FPKI CAs shall:

- include this extension in applicable CA certificates;
- not include this extension in the self-signed certificate;
- set the criticality flag to "true";
- include the permittedSubtrees and excludedSubtrees fields as required;
- only choose the directoryName for the base GeneralName; and
- include the appropriate integer in the minimum and maximum fields of GeneralSubtree to indicate the name space as required.

1.2.12.2 Processing Requirements

FPKI certificate processing entities shall:

- reject the certificate if the nameConstraints extension is flagged critical and the GeneralSubtree base GeneralName in the NameConstraints extension is other than directoryName; and.
- ignore the name forms if the nameConstraints extension is flagged non-critical and the GeneralSubtree base GeneralName in the nameConstraints extension is other than directoryName.

1.2.13 policyConstraints

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

1.2.13.1 Generation Requirements

FPKI CAs shall:

- set the criticality flag to “true”;
- include this extension in applicable CA certificates;
- not include this extension in the self-signed certificates;
- include the requireExplicitPolicy field as applicable; and
- include the inhibitPolicyMapping field as applicable.

1.2.13.2 Processing Requirements

FPKI certificate processing entities shall use the policyConstraints information during the certification path validation process as described in Section 1.4

1.2.14 cRLDistributionPoints

This extension identifies the CRL distribution point or points to which a certificate user should refer to ascertain if the certificate has been revoked. It may be implemented as either a critical or a non-critical extension. This extension provides a mechanism to assemble a CRL that contains certificates revoked for specific reason codes. This extension is used to create a CRL containing certificates revoked for the reasons keyCompromise or cACompromise. For the purposes of this document, this CRL is called an Indirect CRL (ICRL).

1.2.14.1 Generation Requirements

FPKI CAs shall:

- set the criticality flag;
- include this extension in CA and EE certificates;
- not include this extension in self-signed certificates; and
- generate distributionPoints of type directoryName and uniformResourceIdentifier.

1.2.14.2 Processing Requirements

FPKI certificate processing entities shall implement CRL Distribution Point processing as described in Recommendation X.509.

If a certificate processing entity cannot process this extension and the extension is indicated as critical, then the certificate shall be rejected.

FPKI certificate processing entities must process a distributionPoint or type directoryName and uniformResourceIdentifier. Processing of other distributionPoint names is optional. If distributionPoint is absent, the distribution point name defaults to the CRL issuer name.

1.3 Certificate Path Development

Certification path development and path validation procedures work closely together. The most efficient way to develop a path is to start at the end-entity and build a certificate chain back towards the user's trusted CA.

In contrast, as defined by [1], policy constraint, name constraint, signature verification, and parameter-oriented processing must be done in the direction of trust, i.e., from the trusted CA to the end entity. The direction of trust is always the same regardless of how a path is developed: one must find a set of certificates that provide a chain of trust from the trusted CA to the end entity.

Thus, the path development objective is to find a path and provide it to the path validation function. The path validation function must either identify the path as valid, or it must identify where the path fails. The path development module should use that information to attempt to find alternate paths, e.g., search for cross-certificates.

1.3.1 Path Development Procedures

The specific approach to developing a certificate path is an implementer's option. One approach to developing a certification path is provided here. A certification path may be developed, beginning with the subject end entity and terminating at the trusted CA in the following manner, by querying the X.500 Directory for the requisite certificates:

1. Query for the EE signature certificate by creating a certificate request with the following settings:
   - subject DN = end entity DN
   - keyUsage = bit set for digital signature (if signature verification is required), or key encipherment or key agreement (if a PRBAC check or key exchange is required)
   - certificateValid = current date and time in ZULU
   - subjectKeyIdentifier if known (e.g., from the received message)
   - pathToName equal to the end entity name
   - attribute = user certificate[3]

2. Issue the certificate request, and obtain the certificate. If no certificate is available, then stop. Attempt to identify alternative certificate chain, e.g. search for cross-certificates.
3. If the issuer DN in the certificate equals the trusted CA, then stop; path development is complete; else go to step 4.

[3] This attribute may be set to "ca certificate" if subject certificate is that of a CA.

4. Query for the issuer's signature certificate by creating a certificate request with the following settings:
    - subject DN = issuer DN in certificate
    - attribute = ca certificate
    - subjectKeyIdentifier = issuer key identifier in the certificate
    - keyUsage = bit set for digital signature
    - Go to step 2.

Each certificate request may result in one or more certificates being returned to the application. The application will stack the results of each query, but follow only one path at a time. Path discovery may require attempting alternative paths during the path development, or as a result of path validation.

1.4 Certification Path Processing Procedure

The certification path processing procedures for public key certificates is described in detail in Recommendation X.509, Section 12.4. This section provides certification path processing requirements beyond those of Recommendation X.509.

The following inputs to the certification path processing procedure are described in section 12.4 of X.509:

a) an *initial-explicit-policy* indicator value, which indicates if an acceptable policy identifier needs to explicitly appear in the certificate policies extension field of all certificates in the path; *this value shall be set to "true";*

b) an *initial-policy-mapping-inhibit* indicator value, which indicates if policy mapping is forbidden in the certification path; *this value shall be set to "false";*

The following certification path processing checks are in addition to those described in Section 12.4 of X.509, and shall be applied to a certificate:

a) The application must verify that each certificate serial number does not appear on the certificate issuer's CRL (or CRL indicated by CRL Distribution Point extension, especially the ICRL indicated by the CRL Distribution Point with ReasonFlag set to keyCompromise or caCompromise). If the date falls outside the validity interval, the user shall be notified which certificate is outside the interval and when the certificate was, or will be valid, as well as warned that the certificate may have been revoked and given the option to proceed with the data processing. Real-time protocols (e.g. web page access) shall never be permitted to process expired certificates. Note that the PAA certificate validity period is not checked. The notification of an expired certificate (for non-realtime protocols) shall at a minimum :

    1. List all expired certificates in the certificate chain.
    2. Explicitly state that the certificate(s) is/are expired.

3. Explicitly state that the certificate(s) may have been revoked, and that the certificate will not appear on current CRLs even if the certificate has been revoked.
4. Explicitly state that the signature applied to the entity being verified may not be valid. Verification of signatures using expired certificates requires the assistance of the originator's CA.
5. Require positive user action to acknowledge the warning message (i.e. clicking an on-screen OK button) before data processing continues.

b) for an intermediate certificate, the basic constraints extension must be present in the certificate and the **cA** component must be set to true. (Certificates issued by entities without the Basic Constraints Extension present and the cA component set to true shall be rejected - See section 1.2.11.2.) If the **pathLenConstraint** component is present, check that the current certification path does not violate that constraint;

c) Ignore the issuerUniqueIdentifier and subjectUniqueIdentifier fields if present in the base certificate;

d) Check that both the subject and issuer fields of the base certificate contain a distinguished name that is not empty (i.e. must include at least one RelativeDistinguishedName);

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

f) If the keyUsage extension is present and is flagged critical, verify that the keyCertSign bit is set.

2 Version 2 CRLs

CAs use CRLs to revoke a subject's certificate. The CRLs are stored in the directory as attributes and are checked by users to verify that the other users' certificates are not revoked. The fields in a CRL identify the issuer (i.e., CA), the date the current CRL was generated, the date the next CRL will be generated, and the revoked users' certificates. A CA may also add extensions that contain additional information about a specific entry or extensions about the entire CRL (see section 2.1).

As a subset of the CRL, a separate ICRL is generated that contains only certificates with the reason code value of keyCompromise or cACompromise. A single ICRL is issued by a special purpose CA responsible for managing certificates within the Root-CA domain. The ICRL consolidates all "key compromise" CRLs issued by the Root-CAs and CAs within the Root-CA domain (and, optionally, cross-certified domains).

Figure 2-1 illustrates the association between certificate extensions, CRL extensions, and CRL entry extension. The X.509 certificate may contain a CRL Distribution Point extension which includes a field, CRL Issuer. The CRL Distribution Point identifies where to find a CRL for the certificate. The CRL Issuer field identifies the CA that signs the CRL (if other than the issuing CA), and is only applicable if the CRL is an Indirect CRL. The CRL Issuer field must be the same as the Issuer field of the CRL identified by the CRL Distribution Point. Since the CRL is actually an ICRL, the Indirect CRL component of the CRL extension, Issuing Distribution Point, must be set to "true". Finally, the CRL entry extension, Certificate Issuer, may be applied to revoked certificates on an ICRL. This entry extension indicates the CA that issued the revoked certificate. If this extension is omitted in an ICRL, the certificate issuer is "inherited" from the issuer of the previous revoked certificate appearing on the list.

Certificates remain on the CRL and ICRL one inclusion interval beyond their expiration date. The CRL shall use the syntax of the CertificateList as defined in the 1993 X.509 Specification as amended by the Technical Corrigendum (TC for DR128), with CRL and CRL Entry extensions defined in the X.509 Amendment 1. FPKI is using the CertificateList (i.e. V2 CRL) to revoke both user and CA certificates.

CAs may optionally supplement the CRL based revocation mechanisms with on-line revocation mechanisms as specified in [6].

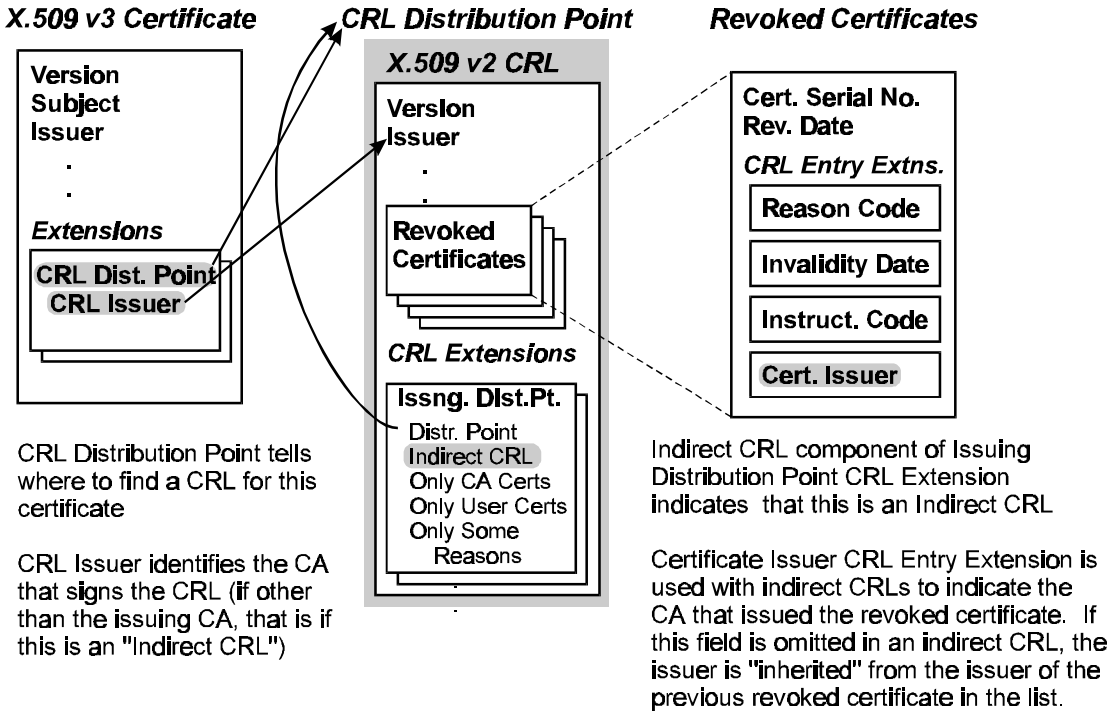

Figure 2-1. Relationship of Certificate, CRL and CRL Entry Extensions

FPKI CAs shall generate and sign CRLs/ICRLs that:

- include the version field to indicate that it is a V2 CRL.
- include the signature field to indicate the algorithm used to certify the CRL (if parameters are associated with the signature algorithm, those parameters shall not be included);
- include use of the Parameterized Type (if parameters are associated with the signature algorithm, those parameters shall not be included);
- include the issuer field to indicate the distinguished name of the CRL issuer;
- include the thisUpdate field to indicate when the CRL was generated;
- include the nextUpdate field to indicate when the next CRL update will be generated, if a scheduled time is known;
- include the revokedCertificates field containing the sequence(s) of userCertificates (which may identify user or CA certificates) field(s), revocationDate field(s), and crlEntryExtensions field(s) to indicate the serial number of each revoked certificate, the time when it was revoked, and the entry extensions (as described in Sections 2.1.4 through 2.1.7); and
- include crlExtensions field(s) as specified in Sections 2.1.1, 2.1.2, 2.1.3. 2.1.8, and 2.1.9.

The following shall apply to both the CRL and ICRL unless stated otherwise. FPKI certificate processing entities shall:

- verify the signature on the CRL/ICRL by employing the public key from the issuer's certificate and parameters, if applicable. If the CRL signature is invalid, the user shall be notified which CRL is invalid and instructed to obtain a new CRL. If a valid CA CRL cannot be obtained, then the user will be given the option to proceed without a valid CRL and without using the invalid CRL;
- verify the certification path of the CRL issuer's signature certificate;
- verify that the version is V2;
- verify the present time falls within the thisUpdate and nextUpdate field(s), if present;
- if present the cRLNumber is present, verify that it is greater than that of the last CRL that the user possesses;
- verify that the CRL issuer is the issuer of the certificate (or as indicated by the CRL Distribution Point extension);
- verify that the subject name in the CRL issuer's X.509 certificate matches the CRL issuer's name and the CRL issuer's certificate BasicConstraints extension CA flag is set to "true".
- if the KeyUsage extension is present in the CRL issuer's certificate and is flagged critical, verify that the keyUsage cRLSign bit is set to 1;
- check whether the certificate serial number appears on the CRL or ICRL. If the certificate appears on either list, and if no reasonCode exists, or a reasonCode exists and is either unspecified, affiliationChanged, superseded, or cessationOfOperation, notify the user of which certificate is on the CRL/ICRL and allow the option to proceed with message processing. If the reasonCode is keyCompromise or certificateHold, the user shall be notified and the certificate shall be rejected thereby halting all associated processing. If a certificate that appears on the CRL/ICRL is a CA certificate, regardless of the reasonCode, the user shall be notified and the certificate shall be rejected thereby halting all associated processing.
- verify that the CA bit is set in the basicConstraints field of the issuer's certificate;
- for ICRLs only, determine if a certificate is revoked by attempting to match the certificate's serial number and issuer with an ICRL entry's serial number and certificateIssuer extension (or value inherited from previous certificateIssuer field as described in the X.509 Draft Amendment 9594-8); and
- for ICRLs only, verify that the cRLIssuer field for the cRLDistributionPoints extension matches the issuer field of the ICRL.

If a current CRL is not available, then the user shall be notified when the CRL was valid, recommended to obtain a new CRL from the Directory, and given the option to proceed with the out of date CRL (signature must still be checked).

The notification at a minimum shall:

- list all expired CRL's in the certificate chain;

- explicitly state that the signature applied to the entity being verified may not be valid; verification of signatures using expired CRL's requires the assistance of the originator's CA;
- require positive user action to acknowledge the warning message (i.e. clicking on an on-screen "OK" button) before message processing continues; and
- If the expired CRL contains the issuingDistributionPoint extension with the reason flags of the onlySomeReasons element set to "keyCompromise" of "cACompromise", then the certificate shall be rejected and message processing halted because the expired CRL is used to list the certificates in which the key has been compromised.

2.1 CRL Extensions

The following sections describe the standard CRL extensions and CRL entry extensions. The CRL extensions add information about the CRL and the CRL issuer, and provide mechanisms to control the size of the CRLs. The CRL entry extensions add information about a specific entry within the CRL.

2.1.1 authorityKeyIdentifier

This non-critical extension identifies the public key to be used to verify the signature on this CRL. It enables distinct keys used by the same CA to be differentiated. This extension may hold the explicit key identifier or an explicit certificate identifier. This extension is useful when a CA uses more than one key (e.g., when the CA key is updated due to crypto-changeover).

FPKI CAs shall:

- include this extension in all CRLs;
- include the authorityKeyIdentifier octet string with a unique identifier for the CA's key material (the subject key identifier of the issuer's certificate); and
- omit the authorityCertIssuer and authorityCertSerialNumber fields.

There are no requirements for FPKI implementations to process this extension, though it is encouraged that CRL processing entities be capable of recognizing the authorityKeyIdentifier field as the identifier of the certification authority's key used to sign the CRL.

2.1.2 issuerAltName

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

FPKI CAs shall:

- set the criticality flag to "false";

- generate this extension as applicable; and
- be capable of populating GeneralName with types dNSName or uniformResourceIdentifier.

FPKI CRL processing entities shall:

- reject any CRL if this extension is present, marked critical and does not contain directorynames; and
- if this extension is present and marked non-critical, it need not contain directorynames.

2.1.3 CRLNumber

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

2.1.4 reasonCode

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

FPKI CAs that generate this extension shall include CRLReason bits for unspecified, key compromise, CA compromise, affiliation change, superseded, and cessation of operation.

There are no requirements for FPKI implementations to perform any automated processing of this extension.

2.1.5 holdInstructionCode

This non-critical CRL entry extension provides for inclusion of a registered instruction identifier to indicate the action to be taken after encountering a held certificate. FPKI CAs and certificate processing entities are not required to support this extension.

2.1.6 invalidityDate

This non-critical CRL entry extension indicates the date at which it is known or suspected that the private key was compromised or that the certificate should otherwise be considered invalid. This date may be earlier than the revocation date in the CRL entry, which is the date at which the CA processed the revocation. FPKI shall use this extension to identify the date at which the certificate should be considered invalid, as per the reasonCode described in Section 2.1.4.

FPKI CAs that generate this extension shall include the date at which the certificate was suspected or known to be invalid.

There are no requirements for FPKI implementations to perform any automated processing of this extension.

2.1.7 certificateIssuer

This critical CRL entry extension identifies the certificate issuer associated with an entry in a CRL which has the indirectCRL indicator set. If this extension is not present on the first entry in an indirect CRL, the certificate issuer defaults to the CRL issuer. On subsequent entries in an indirect CRL, if this extension is not present, the certificate issuer for the entry is the same as that for the preceding entry.

FPKI CAs shall generate this extension only for ICRL entries, and shall include for the GeneralName the directoryName of the certificate issuer as it appears in the "issuer" Name field of the revoked certificate.

2.1.8 issuingDistributionPoint

This critical CRL extension identifies the CRL distribution point for this particular CRL, and indicates if the CRL is limited to revocations for end-entity certificates only, for CA-certificates only, or for a limited set of reasons only. This extension indicates that the CRL may contain entries from CAs other than the authority that signed and issued the CRL.

FPKI CAs shall generate this extension only for ICRLs and shall:

- use the default value of false for both onlyContainsUserCerts and onlyContainsCACerts fields; and
- set the indirectCRLfield to true.

FPKI CRL processing entities shall process this field in accordance with Recommendation X.509.

2.1.9 deltaCRLIndicator

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

The FPKI has no requirements to support this extension.

3 V3 Certificate And V2 CRL Profile

This section lists the protocol elements FPKI CAs must generate to add extensions to the certificate, and the protocol elements FPKI certificate processing entities must understand if the extension is to be processed properly. The lists are presented in tabular format with seven major column headings.

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

The “Protocol Element” column refers to the name of the ASN.1 field taken from the X.500 standards or the X.509 amendment titled “Certificate Extensions.”

The “Proc.” column indicates if processing of the element is mandatory or optional as dictated by the governing policy.

The “Signature Certificate” column specifies the level of support required for each element within a digital signature certificate. The level of support contains the support classifications (see Section 3.1) required by compliant certificate processing entities who process certificates. The “Generation” column is divided into four certificate types: “Root”, “CA”, “EE”, and “Cross”. The information that is to appear in each type of certificate is identified within each respective column. The term “Cross” refers to the profile for cross-certificates. When the CA acts as an End Entity (e.g., when a CA receives a message) then the Proc column applies.

The “KM Cert” column indicates if the extension or attribute is required to be included in Key Management certificates.

The “Notes” column refers to additional information supplied at the end of the table.

3.1 Support Classification

Each of the protocol elements listed in sections 3.2 and 3.3 are designated as having a support requirement of mandatory or optional. Where protocol elements are nested (i.e., the elements contain sub-elements), the requirement to support the nested element is relevant only when the immediately containing (parent) element is supported.

To specify the support level of the certificate extensions the following terminology is defined.

3.1.1 Static Capability

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

mandatory support (**m**) : Federally compliant certificate and CRL generation applications shall be able to generate the protocol element. Federally compliant certificate processing applications shall be able to receive the protocol element and perform all associated procedures (i.e., implying the ability to handle both the syntax and the semantics of the element) as relevant. Populating the information of this protocol element is an implementation detail based on policy decisions.

optional (**o**) : Federally compliant certificate and CRL generation applications are not required to support generation of the protocol element. If support is claimed (i.e., if the optional element is implemented), the element shall be treated as if it were specified as mandatory support, and the sub-elements, if present, shall be supported as specified (i.e., an optional element may have sub-elements indicated as mandatory, "m"; this indicates that if the optional element is implemented, the sub-elements must also be implemented as specified). Federally compliant certificate processing applications shall ignore the protocol element and continue processing the certificate or CRL, unless the element is flagged "critical." See Amendment 9594-8 to X.509 certificates for the rules concerning processing of critical extensions.

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

3.1.2 Dynamic Behavior

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

prohibited (**x**) : Federally compliant certificate and CRL generation applications shall verify that the element is never generated. Federally compliant certificate processing applications will generate and return an appropriate error if a prohibited element is encountered.

critical (**k**) : Federally compliant certificate and CRL processing applications shall, if the element is present in the certificate or CRL and not recognized by the certificate-using system, consider the certificate invalid. The element, if present in a CRL entry and not recognized by the certificate-using system, shall indicate to the user that the CRL may not be as complete as the user expects.

required (r) : the information for this protocol element must be populated upon certificate or CRL generation.

3.2 Base Certificate

Item	Protocol Element	Proc.	Signature Certs.			KM Cert	Notes	Ref.
			Self-Signed	CA	EE			
1.	Certificate	m	mr	mr	mr	mr		
2.	version	m	mr	mr	mr	mr		
3.	serialNumber	m	mr	mr	mr	mr		
4.	signature	m	mr	mr	mr	mr	1	3.2.1/1
5.	issuer	m	mr	mr	mr	mr		
6.	validity	m	mr	mr	mr	mr		
7.	notBefore	m	mr	mr	mr	mr		
8.	notAfter	m	mr	mr	mr	mr		
9.	subject	m	mr	mr	mr	mr		
10.	subjectPublicKeyInfo	m	mr	mr	mr	mr		
11.	algorithm	m	mr	mr	mr	mr	2	3.2.1/1
12.	subjectPublicKey	m	mr	mr	mr	mr		
13.	issuerUniqueIdentifier	o	o	o	o	o	2	
14.	subjectUniqueIdentifier	o	o	o	o	o	2	
15.	extension	m	mr	mr	mr	mr		3.2.2/1

1. Population of the “parameters” field on generation is PROHIBITED. Algorithm parameter processing shall be implemented as described in Appendix A.
2. Should not be populated; must be ignored if values are not understood.

3.2.1 Algorithm Identifier

Item	Protocol Element	Proc.	Signature Certs.			KM Cert	Notes	Ref.
			Self-Signed	CA	EE			
1.	AlgorithmIdentifier							
2.	algorithm	m	mr	mr	mr	mr		
3.	parameters	m	mr	m	m	m		

3.2.2 Extensions

Item	Protocol Element	Proc.	Signature Certs.			KM Cert	Notes	Ref.
			Self-Signed	CA	EE			
1.	Extensions	m	mr	mr	mr	mr		
2.	Extension							
3.	extnID	m	mr	mr	mr	mr		
4.	critical	m	mr	mr	mr	mr		
5.	extnValue	m	mr	mr	mr	mr		

3.2.2.1 Standard Extensions

Item	Protocol Element	Proc.	Signature Certs.			KM Cert	Notes	Ref.
			Self-Signed	CA	EE			
1.	authorityKeyIdentifier	o	o	mr	mr	mr	2	3.2.2.1.1/1
2.	subjectKeyIdentifier	o	mr	mr	mr	mr	2	3.2.2.1.1/15
3.	keyUsage	m	o	kmr	kmr	kmr		3.2.2.1.1/16
4.	extendedKeyUsages	o	o	o	o	o		3.2.2.1.1/26
5.	privateKeyUsagePeriod	o	o	o	o	-		3.2.2.1.1/27
6.	certificatePolicies	mr	o	(k)mr	(k)mr	(k)mr		3.2.2.1.1/30
7.	policyMappings	m	o	m	-	-	1	3.2.2.1.1/37
8.	subjectAltName	m	o	m	m	m		3.2.2.1.1/42
9.	issuerAltName	m	o	m	m	m		3.2.2.1.1/42
10.	subjectDirectoryAttributes	o	o	o	o	o	4	
11.	basicConstraints	m	mr	kmr	kmr	o		3.2.2.1.1/54
12.	nameConstraints	m	o	km	-	-	3	3.2.2.1.1/57
13.	policyConstraints	m	o	km	-	-		3.2.2.1.1/74
14.	cRLDistributionPoints	m	o	(k)m	(k)m	(k)m		3.2.2.1.1/79

1. All cross-certificates are not required to have a policy mapping extension because there is a possibility that no policy mapping is required.
2. Though not mandatory, this extension is recommended for certificate generation and processing.
3. Population of this extension is encouraged to the fullest extent possible.
4. This extension may be used to implement access control as described in SDN.706.

3.2.2.1.1 Standard Extension Syntax

Item	Protocol Element	Proc.	Signature Certs.			KM Cert	Notes	Ref.
			Self-Signed	CA	EE			
1.	AuthorityKeyIdentifier							
2.	keyIdentifier	m	mr	mr	mr	mr	8	
3.	authorityCertIssuer	o	o	o	o	o		
4.	GeneralName							
5.	otherName	o	o	o	o	o		
6.	rfc822Name	o	o	o	o	o		
7.	dNSName	o	o	o	o	o		
8.	x400Address	o	o	o	o	o		
9.	directoryName	m	m	m	m	m		
10.	ediPartyName	o	o	o	o	o		
11.	uniformResourceIdentifier	o	m	m	m	m		
12.	iPAddress	o	o	o	o	o		
13.	registeredID	o	o	o	o	o		
14.	authorityCertSerialNumber	o	o	o	o	o		
15.	SubjectKeyIdentifier	m	mr	mr	mr	mr		
16.	KeyUsage							
17.	digitalSignature	m	m	m	mr	-		
18.	nonRepudiation	m	m	m	m	-		
19.	keyEncipherment	m	-	-	-	m		
20.	dataEncipherment	m	-	-	-	m		
21.	keyAgreement	m	-	-	-	m		
22.	keyCertSign	m	mr	m	-	-		
23.	cRLSign	m	m	m	-	-		
24.	encipherOnly	m	o	o	o	o	9	
25.	decipherOnly	m	o	o	o	o	9	
26.	KeyPurposeId	o	o	o	o	o	9	
27.	PrivateKeyUsagePeriod							
28.	notBefore	m	m	m	m	-	5	
29.	notAfter	m	m	m	m	-	5	
30.	PolicyInformation							
31.	policyIdentifier	m	mr	mr	mr	mr	1	
32.	CertPolicyId							
33.	policyQualifiers	m	m	m	m	m		
34.	PolicyQualifierInfo							
35.	policyQualifierId	m	mr	mr	mr	mr	6,7	
36.	qualifier	m	o	o	o	o		
37.	PolicyMappingsSyntax							
38.	issuerDomainPolicy	m	mr	mr	-	-		

Item	Protocol Element	Proc.	Signature Certs.			KM Cert	Notes	Ref.
			Self-Signed	CA	EE			
39.	CertPolicyId						2	
40.	subjectDomainPolicy	m	-	m	-	-		
41.	CertPolicyId							
42.	GeneralName							
43.	otherName	o	o	o	o	o		
44.	rfc822Name	o	o	o	o	o		
45.	dNSName	o	o	o	o	o		
46.	x400Address	o	o	o	o	o		
47.	directoryName	m	m	m	m	m		
48.	ediPartyName	o	o	o	o	o		
49.	nameAssigner	o	o	o	o	o		
50.	partyName	o	mr	mr	mr	mr	4	
51.	uniformResourceIdentifier	o	m	m	m	m		
52.	iPAddress	o	o	o	o	o		
53.	registeredID	o	o	o	o	o		
54.	BasicConstraintsSyntax							
55.	cA	m	mr	mr	mr	o	d(false)	
56.	pathLenConstraint	m	o	m	-	o		
57.	NameConstraintsSyntax							
58.	permittedSubtrees	m	mr	mr	-	-		
59.	GeneralSubtree							
60.	base	m	mr	mr	-	-	3	
61.	GeneralName							
62.	otherName	o	o	o	o	o		
63.	rfc822Name	o	o	o	o	o		
64.	dNSName	o	o	o	o	o		
65.	x400Address	o	o	o	o	o		
66.	directoryName	m	m	m	m	m		
67.	ediPartyName	o	o	o	o	o		
68.	uniformResource Identifier	o	m	m	m	m		
69.	iPAddress	o	o	o	o	o		
70.	registeredID	o	o	o	o	o		
71.	minimum	m	o	o	-	-	d(0), 2	
72.	maximum	m	o	o	-	-		
73.	excludedSubtrees	m	m	m	-	-		3.2.2.1.1/59
74.	PolicyConstraintsSyntax							
75.	requireExplicitPolicy	m	m	m	-	-		
76.	SkipCerts							
77.	inhibitPolicyMapping	m	m	m	-	-		
78.	SkipCerts							

Item	Protocol Element	Proc.	Signature Certs.			KM Cert	Notes	Ref.
			Self-Signed	CA	EE			
79.	CRLDistPointsSyntax							
80.	distributionPoint	m	o	o	o	o		
81.	DistributionPointName	m	o	o	o	o		
82.	fullName	m	o	o	o	o		
83.	GeneralName							
84.	otherName	o	o	o	o	o		
85.	rfc822Name	o	o	o	o	o		
86.	dNSName	o	o	o	o	o		
87.	x400Address	o	o	o	o	o		
88.	directoryName	m	m	m	m	m		
89.	ediPartyName	o	o	o	o	o		
90.	uniformResource Identifier	m	m	m	m	m		
91.	iPAddress	o	o	o	o	o		
92.	registeredID	o	o	o	o	o		
93.	nameRelativeToCRLIssuer	m	o	o	o	o		
94.	reasons							
95.	ReasonFlags							
96.	unused	o	o	o	o	o		
97.	keyCompromise	m	m	m	m	m		
98.	cACompromIse	m	m	m	m	m		
99.	affiliationChanged	m	o	o	o	o		
100.	superseded	m	o	o	o	o		
101.	cessationOfOperation	m	o	o	o	o		
102.	certificateHold	m	o	o	o	o		
103.	cRLIssuer	m	m	m	m	m		
104.	GeneralName							
105.	otherName	o	o	o	o	o		
106.	rfc822Name	o	o	o	o	o		
107.	dNSName	o	o	o	o	o		
108.	x400Address	o	o	o	o	o		
109.	directoryName	m	m	m	m	m		
110.	ediPartyName	o	o	o	o	o		
111.	uniformResource Identifier	o	m	m	m	m		
112.	iPAddress	o	o	o	o	o		
113.	registeredID	o	o	o	o	o		

Item	Protocol Element	Proc.	Signature Certs.			KM Cert	Notes	Ref.
			Self-Signed	CA	EE			

1. If the requireExplicitPolicy field is present in the policyConstraints extension, this field shall include at least one of the policies applicable to the certificate.
2. The minimum attribute is always required to be present if the extension is included in the certificate.
3. Although the nameConstraints extension is not always required to be present in a certificate, the base attribute is always required to be present if nameConstraints is present.
4. Note that partyName is required to be present if ediPartyName is included in the certificate.
5. One or both of the notBefore and notAfter elements shall be present in this extension.
6. The supported policyQualifier processes are id-pkix-cps and id-pkix-unotice.
7. PolicyQualifierId shall be present if policyQualifierInfo is included in the certificate.
8. If the AuthorityKeyIdentifier is present, then keyIdentifier is required to be present.
9. It is strongly recommended that these key usages not be populated; if these usages are present and the extension is critical, the certificate shall be rejected.

3.3 CRL

Item	Protocol Element	Proc.	Required Support	Notes	Ref.
1.	CertificateList				
2.	version	m	mr		
3.	signature	m	mr		3.2.1/1
4.	issuer	m	mr		
5.	thisUpdate	m	mr		
6.	nextUpdate	m	m		
7.	revokedCertificates	m	mr		
8.	userCertificate	m	mr		3.2/5
9.	revocationDate	m	mr		
10.	crlEntryExtensions	m	mr		3.3.2.1
11.	crlExtensions	m	mr		3.3.1

3.3.1 CRL Extensions

Item	Protocol Element	Proc.	Required Support	Notes	Ref.
1.	authorityKeyIdentifier	o	mr		3.2.2.1/1
2.	issuerAltName	m	m		3.2.2.1/8
3.	cRLNumber	o	o		3.3.1.1/1
4.	issuingDistributionPoint	m	km		3.3.1.1/2
5.	deltaCRLIndicator	o	o		3.3.1.1/8

3.3.1.1 CRL Extension Syntax

Item	Protocol Element	Proc.	Required Support	Notes	Ref.
1.	CRLNumber	m	m		
2.	IssuingDistPointSyntax	m	m		
3.	distributionPoint	m	m		3.2.2.1.1/79
4.	onlyContainsUserCerts	m	m	d(false)	
5.	onlyContainsCACerts	m	m	d(false)	
6.	onlySomeReasons	m	m		3.2.2.1.1/94
7.	indirectCRL	m	m	d(false)	
8.	BaseCRLNumber	m	m	1	
1. The value of this element shall be identical to the value in the cRLNumber extension of the base certificate.					

3.3.2 CRL Entry Extensions

Item	Protocol Element	Proc.	Required Support	Notes	Ref.
1.	reasonCode	o	m		3.3.2.1/1
2.	holdInstructionCode	o	o		
3.	invalidityDate	o	m		
4.	certificateIssuer	m	km		

3.3.2.1 CRL Entry Extension Syntax

Item	Protocol Element	Proc.	Required Support	Notes	Ref.
1.	CRLReason				
2.	unspecified	m	m		
3.	keyCompromise	m	m		
4.	cACompromise	m	m		
5.	affiliationChanged	m	m		
6.	superseded	m	m		
7.	cessationOfOperation	m	m		
8.	certificateHold	m	m		
9.	removeFromCRL	o	o		

APPENDIX A DSA Parameter Processing

The DSA algorithm has optional parameters associated with its syntax which are commonly referred to as *p*, *q*, and *g*. The AlgorithmIdentifier field of subjectPublicKeyInfo is the only place in V3 X.509 certificates in which algorithm parameters will be present. If the DSA algorithm parameters are absent from the subject's DSA X.509 certificate, then the certificate issuer's DSA parameters apply. This strategy validates certificate chains in which all certificates were generated in accordance with the inheritance strategy. The validation strategy will fail validation for certificate chains containing certificates that were generated by certification authorities that did not implement this inheritance strategy.

FPKI CAs shall not populate the parameters in the signature field of the base certificate or in the SIGNED macro in the certificates or the CRLs.

Signatures shall be verified using the algorithm and parameters associated with an "**algorithm state variable**" and a "**parameters state variable**." The following rules describe how these state variables shall be determined and processed during certification path validation:

1. The **algorithm state variable** and **parameters state variable** will be initialized to the *algorithm* and *parameters*, respectively, of the *subjectPublicKeyInfo AlgorithmIdentifier* field of the user's Root-CA signature certificate.
2. For each certificate, the **algorithm state variable** must be equal to the values in the *signature* field and in the *SIGNED* macro. If all three are not equal, the certificate is rejected and another certificate path should be sought. If no alternate certification path can be found, the path validation fails.
3. The *signature* field and *SIGNED* macro will not be populated with parameters in certificates generated by FPKI compliant Cas. If parameters are populated in either of these fields, then they must be verified to be identical to the **parameters state variable**. If a certificate is encountered which has parameters in the *signature* field or the *SIGNED* macro which are not identical to the **parameters state variable**, then the certificate is rejected, and an alternate certificate path should be sought. If no alternate certificate path can be found, the path validation fails.
4. If the **algorithm state variable** is different from the value of *algorithm* in the *subjectPublicKeyInfo* field of the certificate, the **algorithm state variable** will be set to equal the algorithm identifier in the *subjectPublicKeyInfo* field of the certificate, and the **parameters state variable** will be set to "null".
5. If the *subjectPublicKeyInfo* field of the certificate contains public key parameters, the **parameters state variable** will be set to equal the parameters of the *subjectPublicKeyInfo* field of the certificate. The process will be repeated from step 2 onward until the last certificate in the chain is validated.

The public key parameter values at the end of a successful chain validation are the parameters to be used to verify the end entity signatures. Before validating the issuer's signature of a CRL or ICRL, the issuer's certification path must be validated using the certification path validation rules described in steps one through six, above. The CRL issuer's DSA parameters are used as part of the CRL signature verification process. Specifically, any DSA parameters populated in CRL and ICRL signature fields or the SIGNED macros shall never be used as cryptographic algorithm inputs.

APPENDIX B Key Encryption Algorithm Certificate Processing

The parameters *p*, *q*, and *g* will never appear in a FPKI-compliant Key Encryption Algorithm (KEA) certificate. FPKI CAs shall populate the parameters field of the AlgorithmIdentifier within the subjectPublicKeyInfo field of each KEA certificate with an 80-bit parameter identifier (OCTET STRING), also known as the domain identifier. The domain identifier will be computed first by generating a 160-bit SHA-1 hash of the KEA parameters encoded using the DSAParameters syntax. The 160-bit hash will then be reduced to 80-bits by performing an "exclusive or" of the 80 high order bits with the 80 low order bits.

A KEA pairwise key cannot be generated between two users who use public keys generated with different KEA parameters. FPKI compliant certificate using applications have the option of checking the domain identifier for parameter compatibility before attempting to generate a pairwise KEA key, but this check is not required. If a FPKI compliant certificate using application encounters an error when attempting to generate a pairwise key using KEA, then the processing entity may optionally determine if the domain identifiers in the originator's and recipient's KEA X.509 certificate are identical. If they are different, then the application should notify the application user that the KEA key used by the remote participant was generated using incompatible KEA parameters.