NORTEL BNR

A Public Key Infrastructure for U.S. Government Unclassified but Sensitive Applications

Warwick Ford

September 1, 1995

CONTRACT SPONSOR: **National Institute of Standards and Technology**
CONTRACT NO.: **43NANB512705**

Government Contract Sponsor

National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

Contact:
Tim Polk
Tel: (301) 948-3348
Fax: (301) 948-0279

Contractor

NORTEL Federal Systems
2010 Corporate Ridge, Suite 800
McLean, VA 22102

Report author:
Warwick Ford
Bell-Northern Research
Tel: (613) 765-4924
Fax: (613) 765-3520

Contact for contractual issues:
Pat Donahue
Tel: (703) 712-8132
Fax: (703) 712-8982

Summary of Amendments

First draft	1995 June 6	First draft for informal review
Second draft	1995 August 14	Second draft for PKI Steering Committee Technical Working Group distribution
Final report	1995 September 1	

Table of Contents

Table of Contents 2

Abbreviations 4

Executive Summary 6

1. Introduction 8
    1.1 Background 8
    1.2 Purpose and Scope 9
    1.3 Organization of Report 9
    1.4 Acknowledgments 10

2. Assumptions 11
    2.1 Fundamental Policy Objectives 11
    2.2 User Operational Requirements 11
    2.3 User Security Requirements 13
    2.4 General Assumptions 13

3. Services 17
    3.1 Digital Signature Key Management Services 18
    3.2 Confidentiality Key Management Services 19
    3.3 Certificate Management Services 21
    3.4 Directory Services 22
    3.5 End-Entity Initialization Services 23
    3.6 Personal Token Management Services 23
    3.7 Non-repudiation Services 24
    3.8 Client Interface Services 26

4. Functional Description 27
    4.1 Certificate Management 27
    4.2 Infrastructure Architecture 43
    4.3 Naming 50
    4.4 Basic Key Management Functions 53
    4.5 Key, Certificate, and CRL Backup/Archival 60
    4.6 Audit and Alarm 66
    4.7 Distributed Data Management 68
    4.8 Organizational Registration Authority Functions 68

5. Miscellaneous Design Topics 70
    5.1 Interoperation with External PKIs 70

5.2 Communication Protocols 75
5.3 Node Operator Interface 79

References 81

Appendix A — Directory System Requirements 84

Appendix B — Certificate Revocation 88

Abbreviations

ANSI	American National Standards Institute
API	Application program interface
BNR	Bell-Northern Research
CA	Certification authority
CKL	Compromised key list
CMIP	Common management information protocol
COTS	Commercial off-the-shelf
CRL	Certificate revocation list
DAP	(X.500) Directory access protocol
DBMS	Database management system
DISP	(X.500) Directory information shadowing protocol
DIT	(X.500) Directory information tree
DoD	U.S. Department of Defense
DSA	Directory service agent
DSP	(X.500) Directory system protocol
DSS	Digital signature standard
DUA	Directory user agent
EDI	Electronic data interchange
EES	Escrowed Encryption Standard
FIPS	Federal Information Processing Standard
GULS	Generic upper layers security
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IPRA	Internet Policy Registration Authority
ISO	International Organization for Standardization
KEA	Key exchange algorithm
KMID	(MISSI) Key material identifier
LDAP	Lightweight directory access protocol
MISSI	Multi-level Information System Security Initiative

MSP	Message Security Protocol
NIST	National Institute of Standards and Technology
NSA	National Security Agency
ORA	Organizational registration authority
OSI	Open systems interconnection
PC	Personal computer
PCA	Policy certification authority
PCMCIA	Personal Computer Memory Card International Association
PEM	(Internet) Privacy Enhanced Mail
PIN	Personal identification number
PKI	Public Key Infrastructure
RAM	Random access memory
RSA	Rivest, Shamir, Adleman (public-key cryptosystem)
SHA	Secure hash algorithm
SNMP	(Internet) Simple Network Management Protocol

Executive Summary

This report discusses a range of architectural issues relating to the design of a Public Key Infrastructure (PKI) for U.S. federal government use in unclassified but sensitive electronic applications. It is assumed that the PKI will be used for ensuring the authenticity and integrity of sensitive information in electronic transactions and for protecting the confidentiality of sensitive information. Consequently, both digital signature and encryption functions will be supported. Protection of classified information is excluded from consideration.

The PKI will provide a range of services to its clients, including certificate management services, digital signature key management services, confidentiality key management services, directory services, end-entity initialization services, personal token management services, non-repudiation services, and client interface services.

This report, which is based on work performed for the Canadian government, presents several new perspectives and addresses several issues which were not covered in the 1993 report *Public Key Infrastructure Study: Final Report*, prepared by the MITRE Corporation for NIST. In particular, this report assumes use of the X.509 Version 3 certificate format and associated standard extensions, resulting in a more flexible and powerful architecture than was possible with earlier certificate formats. Attention is given to issues of how a federal government PKI would interoperate with PKIs of other national governments, of other tiers of government, and of private industry. Substantial consideration is given to the full set of functions needed in hardware/software implementations of the PKI components and the end-user encryption and digital signature devices they would support.

The architecture proposed involves a structure of certificate management nodes, each administered by a certificate management authority. While there is no fundamental technical requirement to structure the federal PKI as a strict hierarchy, an overall hierarchical structure will be beneficial for the purposes of authority delegation and policy management. The nodes will be operated at various organizational levels within the government, each such node supporting a population of digital signature entities, encryption entities, and/or subordinate management nodes. The PKI nodes maintain and distribute public-key certificates and certificate revocation lists, which enable secure communications to occur between any pair of end entities supported by the PKI. Provision is also made for interoperation with end-user systems (which employ compatible technology) supported by external PKIs operated, for example, by other

national governments or by commercial organizations. Recognized standards, especially X.509, are used as much as possible.

Use of X.509 Version 3 certificates makes it possible to relax many of the restrictions inherent in the recommendations in the MITRE PKI Study, and this report contains several recommendations in this direction.

1. Introduction

1.1 Background

In April 1994, the National Institute of Standards and Technology (NIST) issued the report *Public Key Infrastructure Study: Final Report,* prepared by the MITRE Corporation [NIS1]. This report presented the results of MITRE's study of the alternatives for automating management of public keys and of the associated public key certificates for the federal government.

Throughout the 1993-95 period (i.e., overlapping the period of the MITRE study), Bell-Northern Research (BNR) carried out a series of studies for the Canadian government in support of the development of a Canadian strategy for a public key infrastructure (PKI) to support the protection of unclassified but sensitive information and electronic authorization and authentication (EAA) in Canadian government communication and information processing systems.

While there were various similarities and differences in scope and orientation between the MITRE and BNR studies, three particularly significant differences were:

(a) The BNR studies gave substantially greater attention to issues of how a national government PKI would interoperate with PKIs of other national governments, of other tiers of government, and of private industry.

(b) The BNR studies were not restricted to the assumption that the existing standard certificate formats (i.e., ITU-T X.509 version 1 and 2 formats) could not be extended to accommodate newly recognized needs. On the contrary, the BNR studies recommended the extension of the standard certificate format and contributed to the development of the X.509 version 3 certificate format and associated standard extensions.

(c) The BNR studies addressed, in substantial detail, the full set of functions needed in hardware/software implementations of the PKI components and the end-user encryption and digital signature devices they would support.

Discussions between the U.S. federal government and the Canadian government led to recognition that most of the results of the BNR studies were equally applicable in the U.S. government environment as in Canada. Consequently, it was agreed that BNR produce an adaptation of its studies for the U.S. environment. This report is the result.

1.2 Purpose and Scope

The purpose of this report is to define an architecture for a PKI for U.S. government use for unclassified but sensitive applications, and to describe the functionality of the architecture's components. The PKI will be assumed to support both defacto standard and FIPS approved cryptographic algorithms (the Digital Signature Algorithm FIPS 186, the Secure Hash Algorithm FIPS 180-1, the Data Encryption Standard FIPS 46-2, and the Escrowed Encryption Standard FIPS 185).

1.3 Organization of Report

This report is organized into the following sections:

(a) *Assumptions*: This section identifies basic assumptions regarding U.S. federal government PKI requirements.

(b) *Services*: This section describes *what* the PKI should provide for its clients, their applications, and their equipment, in order to satisfy the requirements identified in (a).

(c) *Functional description*: This section proposes a set of functions to be implemented in the PKI in order to provide the services identified in (b). It includes specific recommendations as to technology to be employed.

(d) *Miscellaneous design topics:* This section addresses sundry topics such as external infrastructure interoperation, communications protocols, and operator interfaces.

The following appendices are included:

(a) *Directory system requirements*: This appendix outlines the requirements of an X.500 Directory System Agent (DSA) which is to support distribution of PKI certificates and CRLs.

(b) *Certificate revocation*: This appendix discusses issues associated with certificate revocation and describes possible certificate revocation approaches.

1.4 Acknowledgments

Acknowledgement is due to the Canadian Communications Security Establishment (CSE) for graciously permitting material from BNR's Canadian Government reports to be reproduced in this report. Appreciation is expressed to Paul Van Oorschot of BNR for his contributions to the original reports.

2. Assumptions

2.1 Fundamental Policy Objectives

It is assumed that the fundamental policy objectives underlying the PKI relate to protection of the security and confidentiality of sensitive information in federal computer systems, in accordance with the Computer Security Act of 1987 [CON1]. This Act defines the term *sensitive information* to mean "any information, the loss, misuse, or unauthorized access to or modification of which could adversely affect the national interest or the conduct of Federal programs, or the privacy to which individuals are entitled under section 552a of title 5, United States Code (the Privacy Act), but which has not been specifically authorized under criteria established by an Executive order or an Act of Congress to be kept secret in the interest of national defense or foreign policy."

It is explicitly assumed that the PKI will be used to provide the following basic forms of protection:

(a) ensuring the authenticity and integrity of sensitive information in electronic transactions (including, in particular, financial transactions);

(b) protection of the confidentiality of sensitive information;

(c) non-repudiation.

This implies that the PKI will support both digital signature and encryption functions.

2.2 User Operational Requirements

User operational requirements are assumed to include:

(1) *Central PKI service facilities*: Some federal departments or agencies seek a centrally operated service to which their users can subscribe.

(2) *Departmental PKI facilities*: As an alternative to (1), larger federal departments or agencies might establish departmental PKI facilities, with limited requirements for support from a central PKI administration. In some cases, a departmental PKI needs to be integrated with its user network and be specially controlled and configured.

(3) *Limited backward compatibility*: In comparatively few federal deprartments or agencies there may be pre-existing digital signature or encryption equipment installed. While the PKI should be backward compatible with existing equipment where this is cost-effective, this study focuses on requirements for new-technology equipment.

(4) *Forward compatibility*: The PKI should be designed for forward compatibility with envisaged future developments such as new cryptographic techniques, digital signature systems, smart cards, EDI, and international cryptographic standards.

(5) *Interoperability within U.S. federal government*: The PKI should support services for digital signatures and for protection of sensitive information within or between federal departments or agencies.

(6) *Interoperability beyond U.S. federal government*: The PKI should support cost-effective external interoperability with public-key infrastructures of the governments of allied countries, financial institutions, industry (U.S. and international), law enforcement agencies, state governments, municipal governments, and commercial service operators providing services for the general public.

(7) *Transparency*: The PKI should be simple to use and should result in no noticeable performance degradation to end-users.

(8) *Responsiveness*: Responses to short term environment changes or other requirements should be user-friendly and tailored to specific user environments.

(9) *Adaptability*: The PKI should be adaptable to meet changing operational and security requirements and environments over time.

(10) *Cost effectiveness*: Costs should be minimized for personnel training, key distribution and accounting human resources, and operations and maintenance. Commercial off-the-shelf (COTS) equipment should be used if possible. Where appropriate, a government-wide approach should be used for purposes of economies of scale.

(11) *Administration*: The PKI interface should be user friendly. There should be client support for such functions as installation, problem solving, and

reconfiguration. Equipment procurement must be easy. Administrative overhead should be minimal.

2.3 User Security Requirements

The primary user security requirements are assumed to be as follows:

(1) *General*: Security mechanisms should be able to evolve as technology evolves.

(2) *Confidentiality*: The confidentiality of sensitive information should be guaranteed. The security of keys should be ensured throughout the entire distribution process. Vulnerability of keys to human intelligence or human error should be minimized.

(3) *Integrity*: The correctness of key material, certificates, and messages should be preserved through their life cycles. Authentication, authorization, and non-repudiation, through digital signatures, should be supported. The trustworthyness of the PKI system itself should be ensured.

(4) *Availability*: The PKI system should be available continually. Availability periods should be tailorable to particular systems or environments. Reaction time to suspected compromise should be minimized. The PKI should be responsive and adaptable to changing requirements and threats.

(5) *Assurance*: An acceptable level of assurance for the entire PKI should be maintained. There should be no increase in vulnerability to individual systems due to connection to the PKI. Residual risk with the PKI should be no greater than with paper based technology.

(6) *Accountability*: The PKI must provide at least the same level of accountability as with paper-based systems. User accounting should be limited to only reporting back exceptional operating conditions. Accounting data should be accessible by designated operators. Accountability should be reflected in an audit trail.

2.4 General Assumptions

The following basic assumptions are made with respect to overall PKI structure:

(a) The PKI is to have distributed functionality. From the administrative perspective, it will be hierarchically structured, in the form of PKI nodes, each of which has been delegated authority to support a sub-community of user systems and,

optionally, to administer a set of subordinate PKI nodes.[1] There is no fundamental requirement to fix the number of hierarchical levels; making the number of levels variable will allow for greater flexibility in serving large or small client organizations.

(b) In general, key generation functions are to be distributed, i.e., placed as low as practical in the hierarchy.

(c) Transaction logging and local audit trail accumulation functions are to be built into systems at all levels of the hierarchy. Provision is also to be made for automatically transferring certain audit records to a superior system.

(d) A government-wide X.500 directory service will evolve as the PKI evolves. This directory service can be used for the distribution of non-sensitive information.

(e) The PKI is to be designed to support latest-technology applications and products which conform to emerging standards. In particular, the following applications/products are targeted:

- electronic mail;
- electronic payment systems;
- secure World-Wide-Web browsers;
- business applications designed for compatibility with X.509-based certification infrastructure standards.

Older technology products can potentially be supported but would generally require some adaptation. Possible adaptation of such products would need to be addressed with product suppliers and is not covered in this report.

(f) Only cryptographic modules which have been validated as meeting FIPS 140-1 [FIPS4] should be used to support the PKI.

(g) Full key life-cycle management needs to be built into the PKI, for all key materials, including all public key pairs throughout the infrastructure.

[1] In general this implies a hierarchical structure of certification authorities. However, such structure is not necessarily strictly hierarchical in the sense that all certification paths start from a hierarchical root and traverse a single tree structure. Certification paths may be structured in different ways if trust requirements so dictate and if appropriate technology is deployed.

The following basic assumptions are made with respect to use of digital signatures for ensuring the integrity and authenticity of electronic transactions:

(a) A range of different technologies may be employed for protecting signing keys and processes, dependent upon the threat and risk assessment of the particular application and environment. Different options include: (1) signing key stored on regular disk storage of PC/workstation, encrypted under a password-derived key; (2) signing key stored in hardware cryptographic processor board of PC/workstation; (3) signing key stored in a personal token (e.g., smart card) but signing process performed on PC/workstation; (4) signing key and signing process embedded in smart card, PCMCIA, or similar device. Only cryptographic modules which have been validated as meeting FIPS 140-1 [FIPS4] should be used.

(b) Names associated with digital signatures should be X.500-based, and should form part of a government-wide X.500 name tree. Certificates should be X.509-based [ISO 9594-8] and an attempt should be made to make the certificate format compatible with other external X.509-based certificate structures, including financial industry digital signature standards [ANSI X9.30, ANSI X9.31] and emerging Internet standards in this area.

(c) Potentially, multiple digital signature algorithms will need to be supported in order to interoperate with all required parties. The primary algorithms assumed are the U.S. Digital Signature Standard (DSS) and Secure Hash Algorithm (SHA) [FIPS1, FIPS2]. Support for the RSA algorithm [RIV1] may also be needed.

(d) The PKI can give a digital signature verifier good confidence that the signature was generated by a particular private key. However, users of digital signatures in particular applications may require further information about a signer, e.g., the authorization privileges of that signer, in order to decide whether or not to accept a signed message. Such information may be conveyed in authorization certificates (also known as attribute certificates), which can indicate the semantic significance of a digital signature to a party verifying such a signature with respect to a particular application. It is assumed that management of authorization certificates for applications which need them is an application, rather than PKI, function.

The following basic assumptions are made with respect to use of encryption for the purposes of confidentiality protection of sensitive information:

(a) Federally approved symmetric cryptographic algorithms will typically be used for encryption of end-user data, with PKI functions providing support for key distribution.

3. Services

This section describes proposed PKI services, i.e., it describes *what* the PKI will provide for its clients, their applications, and their equipment. The services are categorized as follows:

(a) Digital signature key management services;

(b) Confidentiality key management services;

(c) Certificate management services;

(d) Directory services;

(e) End-entity initialization services;

(f) Personal token management services;

(g) Non-repudiation services; and

(h) Client interface services.

The following definitions apply:

(a) *Digital signature entity*: An entity in either or both of the following categories:

- *Signer*, i.e., an accountable entity (either a specific individual person or a person acting in an organizational role) that generates digital signatures;
- *Verifier*, i.e., a person or piece of equipment with a requirement to verify digital signatures.

Use of the term *digital signature entity* may also embrace system components (e.g., PCMCIA token, installed software) associated with a person who acts as a signer or verifier.

(b) *Encryption entity*: A system containing (hardware or software) functionality to support the encryption and/or decryption of sensitive information.

(c) *End entity*: Term used to embrace both encryption entities and digital signature entities, when the same discussion applies to both.

3.1 Digital Signature Key Management Services

Digital signatures operate as follows. A signing entity (signer) generates a digital signature on a data item using a signing key (private key) known only to that entity. A verifying entity (verifier) can validate that digital signature, using a verification key (public key) that can be freely made known to any potential verifier. To ensure the integrity of public keys, a public key is distributed in the form of a public-key certificate, digitally signed by a certification authority.

Digital signatures are used both to support requirements of PKI end-users (e.g., the signing of electronic financial transactions) and to support internal requirements of the PKI (e.g., certificate signing).

The main services provided by the PKI infrastructure, with respect to digital signature key management, are:

(a) ensuring the integrity of the binding between the private key used by a signer and the public key used by verifiers; this depends upon use of one of the following services:

- providing a means to obtain, with high integrity, a public key from a signing entity which generates its own key pairs (the PKI then generates a certificate for that public key for the purposes of further distribution);
- as a less-preferred option[2], providing a service for generating a digital signature key pair and securely transferring the private key to the signing entity.

(b) generating public key certificates and distributing such certificates to verifiers (see separate subsection on certificate management services below);

(c) providing a means for archival of verification keys;

(d) providing support for the recovery of a verification key on request;

(e) providing a means for revocation of previously-distributed public keys, as a result of such events as change in authorization or suspected signing key compromise;

[2] This alternative is not encouraged as it presents weaknesses if a digital signature is to have non-repudiation qualities. Furthermore, it may have business or legal ramifications which preclude its use. It is also expressly forbidden by the ANSI X9.30 and X9.31 standards.

(f) providing seed keys to assist in ensuring that a signing key value used does not have characteristics that might compromise the security of the signature mechanism;

(g) providing security audit trail facilities.

Provision will potentially be made to support multiple algorithms for digital signature, e.g., DSS and RSA algorithms.

3.2 Confidentiality Key Management Services

Provision of confidentiality via encryption operates as follows. An encrypting system encrypts data using an encryption key. Another system (a decrypting system) can only recover the original data if it possesses a corresponding decryption key. It will be assumed that cryptosystems used for encrypting user data are federally approved symmetric cryptosystems, i.e., the encrypting key and decrypting key are the same. Public-key cryptosystems can be used for key distribution purposes, but are generally unsuitable for bulk data encryption for performance reasons.

The primary contribution of the PKI is to ensure that any encrypting system and the corresponding authorized decrypting system(s) for some protected data item possess the same key, but that no unauthorized systems possess that key. In contemporary encryption systems for protecting unclassified information, data encryption keys are commonly generated in the hardware or software units which perform the encryption and/or decryption.

The main services provided by the PKI infrastructure, with respect to confidentiality key management, are:

(a) providing the basis by which encryption entities may authenticate each other, as required to support key distribution processes (such authentication will employ public-key-based digital signature technology); this involves the following service functions:

- establishment of identification information and (optional) generation of key pairs used for authentication;
- updating of authentication key pairs used for authentication;
- distribution of identification information and key pairs used for authentication (see also separate subsection on certificate management services below); and

- revocation of public keys used for authentication.

(b) managing keys to be used for encryption key establishment (such key establishment processes will employ public-key technology)[3]; this involves the following service functions:

- (optional) generation of key establishment keys;
- updating of key establishment keys;
- distribution of key establishment keys (see also separate subsection on certificate management services below); and
- revocation of key establishment keys.

(c) assisting in the making of authorization decisions as to whether or not a decryption key should be distributed to a particular system;

(d) providing a means for (short term) back-up or (long term) archival of key establishment keys which, in some circumstances, enable decryption keys to be recovered[4];

(e) providing support for the recovery of a decryption key in various circumstances, such as:

- on client request, in the event of key loss owing to such events as equipment failure or forgotten password;
- on request of authorized management personnel in the client's organization;
- on request of law enforcement agencies with appropriate authorization.

(f) providing random seeds for use in initializing pseudorandom number generators; this can assist in ensuring that a key value used for data encryption, authentication, or key establishment does not have characteristics that might compromise the security of the cryptographic mechanism;

(g) providing security audit trail facilities.

---

[3] In some circumstances, one public-key cryptosystem key pair can serve both purposes of authentication and key establishment, but this is frequently not advisable. This study accommodates the more general case where distinct keys are used for these purposes.

[4] When a long-term key pair is used as the basis of encryption key distribution (e.g., use of RSA to encrypt a copy of an encryption key to accompany an encrypted electronic mail message), then backup of the private key of the key pair can provide a basis for recovery of decryption keys. In key management approaches that do not employ long-term keys (e.g., on-line Diffie-Hellman key negotiation on a per-session basis) such recovery cannot be provided.

Provision will be made to support multiple algorithms for:

(a) data encryption (e.g., DES, EES, other approved algorithms);

(b) digital signatures for authentication purposes (e.g., DSS, RSA);

(c) key establishment (e.g., KEA, RSA, Diffie-Hellman).

3.3 Certificate Management Services

Certificate management services are used to support both digital signature key management and confidentiality key management. The main certificate management services provided by the PKI infrastructure are:

(a) generation of public-key certificates for signers that participate in the digital signature system;

(b) generation of public-key certificates for use in authenticating encryption entities and/or in establishing encryption keys for use between communicating encryption entities;

(c) generation of public-key certificates for certification authorities, signed by other certification authorities, for use in verifying certificate chains;

(d) generation of certificate revocation lists (CRLs); a CRL is a signed list of the certificates which have been revoked by a certification authority; the list is signed by that certification authority;

(e) posting certificates from (a), (b), (c), and (d) for distribution, via Directory Services, to any system requiring them;

(f) providing a means for an entity to initially obtain, and to maintain, a reliable copy of a public key to be used as a starting point in verifying certificate signatures (potentially in a certificate chain) (see also *End-Entity Initialization Services*);

(g) providing facilities as needed to support interoperation between the federal government PKI and external PKIs (operated by other governments or industry);

(h) providing security audit trail facilities associated with certificate generation and revocation.

3.4 Directory Services

Directory services constitute a primary means of distributing certificates and other information regarding people and functional components that use or form part of the PKI. These services are used to support both digital signature key management and confidentiality key management. Directory services employ a distributed (as opposed to centralized) directory system.

The main directory-related services provided by the PKI infrastructure are:

(a) maintaining directory entries for encryption entities; information included in such an entry might include, for example, public-key certificate(s), network address, and contact information;

(b) maintaining directory entries for signers that participate in the digital signature system; information included in such an entry might include, for example, public-key certificate and contact information;

(c) maintaining directory entries for certification authorities; information included in such an entry will typically include public-key certificates with that certification authority as the subject, public-key certificates for other certification authorities issued by that certification authority, and certificate revocation lists (CRLs) issued by that certification authority;

(d) ensuring that unique names exist for all objects (including encryption entities, digital signature entities, and certification authorities) in the PKI. A single name may be used for an entity which has both encryption and digital signature capabilities;

(e) providing servers that make confidential directory information available to authorized persons/systems within the PKI environment (this does not apply to certificates and CRLs but might apply to administrative information held in the directory, e.g., entries for personnel operating PKI services);

(f) delivering non-confidential directory information to external directory servers used by PKI entities and/or entities persons/systems outside the PKI environment;

(g) maintaining access control information, and enforcing access control, to ensure that only properly authorized persons or systems can read information in the directory entries (a), (b), and (c), and that only properly authorized persons can create or modify such entries;

(h) providing access as needed to external directory services used in supporting interoperation between the PKI public-key infrastructure and external public-key infrastructures (operated by other governments or industry).

It should be noted that the sensitivity of attributes in directory entries varies. Some directory information, such as device location and communication address information, may be sensitive and needs to be held in trusted directory systems integrated into the PKI. For this sensitive information, directory access controls, in conjunction with strong authentication, need to be applied to the reading and writing of the directory attributes.

Other directory information, in particular, public-key certificates and CRLs are generally not sensitive. (The digital signatures of certificates protect their integrity.) This information can be distributed by low assurance "open" X.500 directory services which do not form part of the PKI per se. In order to minimize costs, use of such services is recommended as much as possible. For public-key certificates and CRLs, reasonable precautions should be applied to avoid posting of false certificates or CRLs, but this is not a critical security issue as the user of a certificate or CRL will always detect a false certificate or CRL. Hence, directory access controls are not critical and strong authentication mechanisms are not essential. Nevertheless, these features are recommended, to prevent against denial-of-service attacks.

3.5 End-Entity Initialization Services

Systems, devices, or software used for confidentiality or digital signature purposes generally require a special process to be invoked in conjunction with their installation or initialization, in order to establish initial resident key material. Such a process typically involves both on-line electronic exchanges with a PKI system and an activity involving manual intervention, such as insertion of special key material media, manual entry of initial key material, or operation of key fill equipment. This may involve the use of personal tokens as discussed in 3.6.

The main services provided by the PKI infrastructure for end-entity initialization are:

(a) supplying physical media/devices or initial key material required for a manual intervention activity;

(b) conducting required on-line electronic exchanges to complete the establishment of initial resident key material for a component.

3.6 Personal Token Management Services

A personal token is an optional PKI component in the form of a small electronic device which is issued to and can be carried by a person. A personal token is used to associate a particular key (or set of keys), authorization privilege information, and/or cryptographic processing capability with a particular person. A personal token is typically realized as a

smart card or PCMCIA card. One physical token (e.g., smart card) can typically support multiple applications, so is not necessarily dedicated to PKI purposes.

The main services provided by the PKI infrastructure, with respect to management of personal tokens, are:

(a) establishing private keys securely within a personal token for use in generating digital signatures and/or in supporting confidentiality key management (the corresponding public key(s) need to be made known to the PKI infrastructure);

(b) loading a personal token with a public key of a certification authority;

(c) loading a personal token with PKI authorization privilege information;

(d) loading a personal token with cryptographic software/firmware of adequate assurance;

(e) maintaining a security audit trail of personal token initialization activities.

3.7 Non-repudiation Services

Non-repudiation involves the generation, accumulation, retrieval, and interpretation of evidence that a particular party processed a particular data item. The evidence must be capable of convincing an independent third party, potentially at a much later time, as to the validity of a claim. Digital signatures are an essential tool used in providing non-repudiation services, but the existence of a digital signature infrastructure does not, in itself, satisfy all non-repudiation requirements.

Beyond basic digital signature support, the following services to support non-repudiation can be provided by the PKI infrastructure to those users requiring them:

(a) storing of evidence to assist in possible future dispute resolution;

(b) evidence retrieval and interpretation, in the event of dispute situations;

(c) time-stamping of electronic transactions, i.e., affixing of a high-assurance signed time-stamp, as evidence as to the time at which a transaction was signed or communicated.

In the absence of a trusted time-stamping service, provided by a trusted third party or a trusted clock in a hardware cryptographic token, note that:

(a) signature verification (and non-repudiation services) will be possible only for the duration of the validity period of signature key certificates (typically 2-5 years);

(b) if a signature key pair is revoked owing to suspected key compromise, all previous signatures created using that key pair, including those by the authorized user, become open to question (i.e., a successful cryptographic verification of the signature is not adequate to establish validity of the signature).

If either of the above conditions is critical, e.g., if (selected) signatures must be verifiable beyond the original validity period of the public key of a signature key pair, then a trusted time-stamp service is essential. Regarding the first condition, if it is required that some (hopefully a small subset of) signatures be verifiable well into the future (including beyond the expiry date of the signature public-key certificate), then such signatures could be taken to a trusted third party, which:

(a) verifies the signature at that point in time; and

(b) countersigns after appending a trusted time-stamp (the time-stamp is trusted since the trusted third party is).

If the trusted time-stamping technique is also cryptographic, then the trusted third party should continually oversign before the end of each expiry period (of the trusted third party's own signature verification key) to maintain notarization of the signature.

In the absence of a trusted time-stamp service, it remains a possibility that a user may deliberately reveal his own signature key and claim it to be compromised, in order to intentionally repudiate all previous signatures.

Because of the negligible probability (in the absence of undiscovered key compromise) of a dishonest party successfully cheating in a non-repudiation dispute, non-repudiation disputes are expected to be rare (more so than even current handwritten signature disputes). For practical and economic reasons, it is therefore envisioned that a fully-automated process will not be required. Rather, interactive search and retrieval processes, supported by appropriate software tools provided with certificate management node software, will be used at those certificate management nodes which support long-term archives to assemble the required evidence to resolve disputes. Such retrieval of evidence and dispute resolution could be carried out by an appropriately trained (human) trusted third party, such as a government security officer, whose availability would be required on a non-priority basis. (Immediate availability is not essential, as same-day dispute resolution should be no more a requirement than for resolution of analogous disputes involving hand-written signatures currently.)

3.8 Client Interface Services

Operation of the PKI will depend upon interactions between client personnel and operations and administration personnel at certificate management nodes. The main client interface services which may be provided by the PKI are:

(a) registering, de-registering, and processing directory information changes for encryption entities and digital signature entities;

(b) registering, de-registering, and assigning privileges to certificate management node personnel (this will be handled locally by each certificate management node, with respect to personnel associated with that node);

(c) supporting end-entity initialization (see also *End-Entity Initialization Services* above);

(d) performing initialization of personal tokens with respect to data used by the PKI;

(e) processing requests for recovery of decryption key information (in the event of key loss owing to such events as equipment failure or forgotten password);

(f) processing requests for public-key certificate revocation (as a result of such events as change in authorization or suspected signing key compromise);

(g) processing evidence retrieval or arbitration requests associated with the non-repudiation services;

(h) operating a problem desk;

(i) providing training to PKI user personnel;

(j) providing guidance on policy and standards issues.

4. Functional Description

4.1 Certificate Management

When a verifier entity or an encryption entity (a public-key user) needs to use a public key of a remote signer entity or encryption entity (a public-key owner), the public-key user 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 certification authority 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 signed by one certification authority, and zero or more additional certificates of certification authorities signed by other certification authorities.

This subsection addresses the management of public-key certificates used to support digital signature verification or the identification and authentication needs of encryption key distribution. This includes the posting of certificates to external Directory services and the revocation of certificates.

4.1.1 Public-Key Certificate Format (Versions 1-2)

The most widely recognized standard public-key certificate format is that defined in the Directory Authentication Framework X.509 [ISO/IEC 9594-8]. The X.509 certificate format has evolved through three versions — the 1988 version (v1), the 1993 version (v2), and a new version (v3) which has recently been specified by ISO/IEC/ITU and by ANSI X9. X.509 v3, which is described in 4.1.2, allows for many certificate extension fields required for the PKI, and it is recommended that PKI planning assume use of v3. For completeness, this section describes the earlier versions[5].

[5] MISSI currently assumes use of the v2 format.

The X.509 v2 (1993) public-key certificate in ASN.1 notation [ISO/IEC 8824] is as follows:

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

Version ::= INTEGER {v1 (0), v2 (1)}
CertificateSerialNumber ::= INTEGER
Validity ::=
SEQUENCE {
             notBefore    UTCTime,
             notAfter     UTCTime}

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

AlgorithmIdentifier ::=
SEQUENCE {
             algorithm    ALGORITHM.&id({SupportedAlgorithms}),
             parameters   ALGORITHM.&Type({SupportedAlgorithms}
                          {@algorithm}) OPTIONAL }
 -- Definition of the following information object set is deferred, perhaps
 -- to standardized profiles or to protocol implementation conformance
 -- statements.  The set is required to specify a table constraint on the
 -- parameters component of AlgorithmIdentifier.
 -- SupportedAlgorithms   ALGORITHM ::= {...|...}
```

The fields are interpreted as follows:

(a) *Version*: Indicator of the particular certificate format, allowing for future revisions. The format defined in the 1993 X.509 standard is version 2.

(b) *Serial number*: Unique number for this certificate, assigned by the issuing certification authority.

(c) *Signature*: Algorithm identifier of the signature algorithm used to sign the certificate.

(d) *Issuer*: X.500 name of the issuing certification authority.

(e) *Validity*: Start and expiry date/times for the certificate.

(f) *Subject*: X.500 name of the entity holding the private key, for which the corresponding public key is being certified.

(g) *Subject public-key information*: An algorithm identifier plus a public key value for the subject.

(h) *Issuer unique identifier*: An optional bit string field providing additional identification information on the issuing certification authority.

(i) *Subject unique identifier*: An optional bit string field providing additional subject identification information.

Provision is made for one identified entity (subject) to have multiple public keys, and corresponding public-key certificates, for different purposes and/or different algorithms. For example (dependent upon PKI algorithm support policy), an entity might have distinct public keys for:

(a) digital signature, using DSS with the SHA hash function;

(b) digital signature, using RSA with the MD5 hash function;

(c) key establishment, using the KEA algorithm.

When an entity has multiple active key pairs and corresponding certificates, the correct certificate for a particular purpose is indicated by the algorithm identifier in the *subject public-key information* field. For example, for the cases (a), (b), and (c) above, different algorithm identifiers are defined for *DSS-with-SHA*, *RSA-with-MD5* and *KEA* respectively.

4.1.2 Public-Key Certificate Format (Version 3)

The v3 certificate extends the v2 format, by providing for extension fields in a certificate. Particular extension fields may be specified in standards or may be defined and registered by any organization or community having a need.

The X.509 v3 public-key certificate [ISO/IEC DTC] in ASN.1 notation is:

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

Version ::= INTEGER { v1(0), v2(1), v3(2) }
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
&ExtnType
                           -- for the extension object identified by extnId --
}
```

The fields are the same as for the v2 certificate, except for the extensions field which allows addition of new fields to the structure without modification to the ASN.1 definition. An extension field consists of an extension identifier, a criticality flag, and a canonical encoding of a data value of an ASN.1 type associated with the identified extension. When an implementation processing a certificate does not recognize an extension, if the criticality flag is FALSE, it may ignore that extension. If the criticality flag is TRUE, unrecognized extensions cause the structure to be considered invalid, i.e., in a certificate, an unrecognized critical extension would cause validation of a signature using that certificate to fail.

The following ASN.1 information object class[6] is used to define specific extensions:

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

[6] The ASN.1 information object class feature is defined in Part 2 of [ISO/IEC 8824].

ISO/IEC and ANSI X9 are also specifying a set of standard extensions for use in the v3 extensions field [ISO/IEC DAM]. These extensions, which are separated into three categories, are as follows:

a) **Key and Policy Information:** These extensions convey additional information about the subject and issuer keys, such as key identifiers and indicators of approved key usage. They also convey indicators of certificate policy. They facilitate the implementation of public-key infrastructures and allow administrators to limit the purposes for which certificates and certified keys are used.

i) *Authority Key Identifier*: This field enables distinct keys used by the same certification authority to be differentiated (e.g., as key updating occurs). The key may be identified by an explicit key identifier, by identification of a certificate for the key (using certificate issuer name and certificate serial number), or both. This field is important for the PKI, in order for certificate-using products to be able to efficiently find correct certificate chains, taking into account the need for regular key pair updating as part of key life cycle management.

ii) *Key Attributes*: This field provides for optional additional information about the public key being certified. It can include a key identifier, an indication of the intended use of that key (without the requirement that use must be restricted to this purpose), and/or an indication of the valid period of use of the corresponding private key. The key identifier component of this field is important to the PKI in order for certificate-using products to be able to efficiently find correct certificate chains; it can also be used to convey a MISSI Key Material Identifier (KMID). The component indicating valid period of use may also prove important in the long term in the support of trusted third party services.

iii) *Certificate Policies*: This field lists certificate policies that the certificate is expressly recognized as supporting, together with optional qualifier information pertaining to these policies. This field is very important for the PKI, as it is an essential field for large public-key infrastructures which support multiple policies. It will likely be used in all PKI certificates. Using this field, one CA may support multiple policies. In each certificate the issuing CA indicates for which policy or policies that certificate may be used. As of the July 1995 editing of the standard extensions specification, a certificate policy may be composed of a set of separately-registered certificate policy elements.

iv) *Key Usage Restriction*: This field indicates a restriction imposed as to the purposes for which, and policies under which, the certified public key may be used. This field is unlikely to prove important to the PKI, because key usage can generally be controlled adequately through the use of certificate policies.

v) *Policy Mappings*: This field, which is for use in CA-certificates only, allows a certificate issuer to indicate that one or more of that issuer's policies can be considered equivalent to another policy used in the subject CA's domain. This

field may find limited use in the PKI in the cross-certifying of external certification authorities, e.g., in allied governments. As of the July 1995 editing of the standard extensions specification, a certificate policy may be composed of a set of separately-registered certificate policy elements, and policy mapping maps any set of one or more policy elements to any other set of such elements.

b) **Subject and Issuer Attributes:** These extensions support alternative names for certificate subject and issuer. They can also convey additional attribute information about the subject to assist a certificate user in being confident that the certificate applies to a particular person or device.

i) *Subject Alternative Name*: This field contains one or more alternative names, using any of a variety of name forms, which are bound by the certification authority to the certified public key. This field is important to the PKI in order to be able to support certain applications, such as electronic mail or EDI, which may employ their own name forms and not be fully integrated with X.500 directories.

ii) *Issuer Alternative Name*: This field contains one or more alternative names, using any of a variety of name forms, for the certificate issuer. This field may be important for the PKI in supporting certain applications which are not fully integrated with X.500 directories, e.g., for retrieving certificates or CRLs via electronic mail.

iii) *Subject Directory Attributes*: This field conveys any desired X.500 attribute values for the subject of the certificate. This field is very important for the PKI, in order to convey clearance and other subsidiary identification information for PKI subjects. One important potential use is the dissemination of clearance information, if clearances are to be supported by the PKI (see 5.1.4). It would be necessary for the U.S. government to define an appropriate X.500 attribute for clearance information for subjects.

c) **Certification Path Constraints:** These extensions allow constraint specifications to be included in CA-certificates, i.e., certificates for CAs issued by other CAs, to facilitate the automated processing of certificate chains by certificate-using systems when multiple security policies are involved, e.g., when policies vary for different applications within an environment or when interoperation with external environments occurs. The constraints may restrict the types of certificates which can be issued by the subject CA or which may occur subsequently in a certificate chain.

i) *Basic Constraints*: This field indicates whether the certified subject may act as a certification authority, an end entity, or both. If the subject may act as a certification authority, a certification path length constraint may also be specified,

as may a subtrees constraint giving a set of subtree root names; all subsequent subject names in the certification path must fall within one of the identified subtrees. The basic constraints field is important to the PKI in order to protect against fraudulent certificate generation by end-users and to enable U.S. government certificate-using systems to check that external certification authorities are not behaving in an unauthorized manner. The path length constraint and subtrees constraint may be used by the PKI when cross-certifying external certification authorities, in order to control the types of chains generated by such external certification authorities. The subtrees constraint will likely prove adequate for constraining name spaces in the PKI, although it is possible that the name constraints extension (described below) may prove necessary for some purposes.

ii) *Name Constraints*: This field specifies a set of constraints with respect to the names for which subsequent certification authorities in a chain may issue certificates. This field, which is substantially more complex to implement and administer than the subtrees constraint, may prove useful in the PKI when certifying external certification authorities in commercial or other national government infrastructures. If this extension field is used in the PKI, both the name space constraint and the subordinate-to-CAs-superior name subordination options (see 4.3) should be implemented.

iii) *Policy Constraints*: This field specifies a set of constraints with respect to explicit certificate policy identification and policy-mapping restrictions. This field is expected to be needed in the PKI when managing complex external chains.

The Key Usage Restrictions, Name Constraints, and Policy Constraints extensions are always critical, and the Basic Constraints extension may be either critical or non-critical, at the option of the CA generating the certificate. All other extensions are always non-critical.

Migration of existing applications, e.g., X.400 or X.500 applications designed to use the X.509 v1 or v2 format, to be able to use the v3 format will require careful management. In particular, a migration program for MISSI will need to be put in place. Apart from MISSI, such applications are not in widespread use, so migration problems are minimal. Once the basic v3 format enters general use, further migration to the adoption of particular certificate extensions can be achieved in a controlled manner provided the extensions are non-critical. Critical extensions should only be used where they are absolutely essential.

Information conveyed in certificates is not generally subjected to any confidentiality controls. Therefore, it will be necessary for PKI administrators to carefully monitor the use of certificate fields to ensure that sensitive information is not disclosed through the use of these fields.

4.1.3 Public-Key Certificate Generation

A certification authority issues public-key certificates for its subjects. A subject may be a digital signature signer entity, encryption entity, or another certification authority. The issuing certification authority must:

(a) have been assigned administrative responsibility for the subject digital signature entity, an encryption entity, or certification authority; and

(b) have assurance that, as a result of a past or future end-entity initialization or key-pair updating process, the public key being certified is associated with a private key that has been installed, or will be installed, in the identified subject end-entity.

Most commonly, certificate generation will occur in conjunction with public-private key-pair generation (see 4.4.3) and either end-entity installation (see 4.4.4) or public-private key-pair updating (see 4.4.5).

Sometimes, certificate generation may not correspond to introduction of a new public-private key pair. This can occur, for example, when a certification authority key is updated, or when a new certification authority takes over responsibility for an existing encryption entity, digital signature signer entity, or certification authority.

4.1.4 Certificate Distribution

Certificates need to be distributed to:

(a) digital signature verifier entities, which use such certificates to obtain reliable copies of public keys for use in verifying digital signatures; and

(b) encryption entities, which use such certificates to obtain reliable copies of public keys for use in authenticating and/or establishing encryption keys with other encryption entities.

Distribution of certificates to parties requiring them does not require confidentiality protection, nor does it require integrity protection beyond that inherent to a certificate by virtue of it being signed by a trusted authority. In this report, communications systems adequate for distributing certificates are therefore termed "low assurance" communications systems.

Methods of certificate distribution include:

(a) Certificates are posted in a directory for retrieval as needed by any entity. Given the use of X.500 naming in certificates, any X.500 directory can serve this purpose.

(b) Certificates are communicated in application protocols. For example, a protocol conveying a signed data item may provide for also conveying the certificate of the signer's public key plus, to the extent the certification path is known, other certificates forming a certificate chain.

Design of the PKI should provide for both distribution methods as neither can, on its own, satisfy all distribution requirements.

With regard to method (a), it is proposed that PKI certificates be distributed via X.500 systems comprising Directory Service Agents (DSAs) operated by any of:

(a) Federal departments or agencies; these DSAs would hold entries which contain certificates for the department's own end entities (and possibly CA certificates).

(b) A federal infrastructure service provider (e.g., the General Services Administration); these DSAs would hold entries which contain certificates for end entities (and possibly CA certificates) of federal departments or agencies who do not have their own X.500 systems.

(c) The PKI operating authorities; these DSAs would hold entries for certification authorities and for particularly sensitive end entities (while certificates are not generally sensitive, other attributes in a directory entry may be highly sensitive).

In general, access controls are applied to prevent unauthorized access to sensitive attributes in directory entries (public-key certificates are not, in general, sensitive). The non-sensitive parts of sensitive government departmental or PKI entries (including those of certification authorities) can also be replicated to low assurance open X.500 systems (e.g., operated by a federal infrastructure service provider) providing an economical means of access by widely-distributed end entities. The replication would be accomplished via a controlled process which executes periodically, preferably using the X.500 Directory Information Shadowing Protocol.

Any certification authority should be authorized to directly post the certificates and CRLs it generates in the directory system.

Appendix A describes the requirements of X.500 DSA implementations used to support certificate distribution for the PKI.

4.1.5 Certificate Revocation

A certificate includes a validity period. Certificate revocation results from one of the following occurring prior to the expiry date/time:

(a) The public key is no longer considered valid, e.g., because of suspected compromise of the corresponding private key.

(b) The subject entity identified in the certificate is no longer considered an authorized user of the corresponding private key, e.g., because a person has changed his or her organizational affiliation, or because authorized privileges have changed.

(c) Other information in the certificate has changed, e.g., some component of the subject's name or some attribute in the certificate has changed.

Revocations can be notified to public-key users in various ways — Appendix B describes the approaches available. A revocation approach can be categorized as either an off-line or on-line approach. The off-line approach involves a certification authority regularly posting, via a directory service, a signed list of revoked certificates called a certificate revocation list (CRL). A CRL can be distributed via a low assurance directory service, in the same way as public-key certificates (see 4.1.4). The integrity of a CRL's contents, including the binding to a specific date/time of issue, can be guaranteed through the digital signature. Assuming a certification authority is known to post its latest CRL on a periodic basis (e.g., daily) a using entity can always know whether it has the latest list. However, a certification authority cannot reliably advise of revocations on a finer time granularity than its regular period. Even if it posts a new CRL earlier than its regular posting, the CRL user would not be able to detect an attack involving deletion of such a CRL from the low assurance directory. The off-line revocation approach is supported by published standards (ISO/IEC 9594-8), and is being used by other external public-key infrastructures, including PEM.

In an on-line revocation approach, certification authorities are not constrained by the time granularity of a periodic CRL posting. Revocation information can be posted at any time. A public-key user executes an on-line exchange to obtain the latest revocation information. It is possible to format on-line revocation information the same as an off-line CRL and implement the on-line exchange as an X.500 directory access operation. However, the request must be serviced by a directory server (DSA) which is integral to the PKI and which is trusted to ensure that directory information cannot be altered by unauthorized persons. Furthermore, transactions between end entities and such DSA must either be conveyed over secured communications facilities or be implemented as a signed directory operation. While alternative protocols for on-line revocation might be devised, this appears unnecessary for PKI purposes.

Because of the costs associated with implementing an on-line revocation approach, it is recommended that the PKI adopt the off-line approach as its primary revocation approach. Based on a threat and risk assessment, a suitable CRL posting period will need to be determined for each certification authority. If the v2 CRL format (see 4.1.6) is employed, separate lists can be issued for routine revocations and for compromise situations, and a higher-frequency posting period can be used for the latter (e.g., daily posting of routine revocations; hourly posting of compromise revocations). Recognizing that the PKI may be required to support applications for which timeliness of revocation posting is particularly critical, an on-line revocation facility will be considered to be a PKI option. The latter facility will be assumed to employ trusted X.500 servers with signed operations. It will be possible to selectively use the on-line revocation facility, based on the needs of particular applications and/or particular users.

Regardless of off-line or on-line approach, the following CRL-related procedures are assumed. CRLs are distributed via a directory service; the latest CRL from a particular certification authority is held as an attribute in a directory entry associated with that certification authority. When a certificate is revoked, its serial number is added to the CRL. When the expiry date of a certificate on the CRL is reached, that certificate remains on the list for one more CRL issue only, then is removed from the CRL. Each certification authority would typically post its own CRLs in the directory.

A public-key user entity validating any public key needs to decide whether or not it needs to check CRLs. This decision is application-dependent. It is the responsibility of the PKI to ensure that CRLs are available via a directory service if needed.

When a certification authority issues certificates for both end-entities and other CAs, provision should be made for having separate CRLs for end-entities and for CAs stored in separate X.500 attributes (the CertificateRevocationList attribute and AuthorityRevocationList attribute respectively, in accordance with X.509 [ISO/IEC 9594-8]). This separation is important for ensuring efficient performance in processing certificate chains, since the CRL for CA certificates will generally be very small. However, the 1988 and 1993 versions of X.509 contained a flaw, in that it was impossible to securely distinguish between the two CRLs thereby making it impossible to detect unauthorized substitution of one for the other. Therefore, this feature should not be implemented without also implementing the IssuingDistributionPoint CRL extension described in the following subsection.

4.1.6 Certificate Revocation List Format

The CRL format which is most widely used currently is that defined in the 1993 revision of X.509 [ISO/IEC 9594-8], which is identical to the PEM format. This format obsoleted the 1988 format. However, as with certificate format, a new CRL format which includes extension fields has recently been specified by ISO/IEC and ANSI X9. Because

the new format allows for certain certificate extension fields required for the PKI, it is recommended that PKI planning assume use of the new format.

The 1993 format, in ASN.1 notation [ISO/IEC 8824] is as follows:

```
CertificateList ::= SIGNED { SEQUENCE {
       signature                 AlgorithmIdentifier,
       issuer                    Name,
       thisUpdate                UTCTime,
       nextUpdate                UTCTime OPTIONAL,
       revokedCertificates       SEQUENCE OF SEQUENCE {
             userCertificate     CertificateSerialNumber,
             revocationDate      UTCTime} OPTIONAL }}
```

The fields are interpreted as follows:

(a) *Signature*: Indicator of the algorithm used in signing this CRL.

(b) *Issuer*: Name of the certification authority whose certificates are being revoked and whose signature appears on this CRL.

(c) *This Update*: Date of issue of this CRL.

(d) *Next Update*: Date of issue of next CRL (this is optional in X.509, but mandatory in PEM; therefore its use is recommended).

(e) *User Certificate*: Certificate serial number of a revoked certificate.

(f) *Revocation Date*: Effective date of revocation of a particular certificate.

The new CRL format, which is called the v2 format, has the following ASN.1 notation [ISO/IEC 8824]:

```
CertificateList ::= SIGNED { SEQUENCE {
version                   Version OPTIONAL,
                                -- If present, version must be v2--
       signature                AlgorithmIdentifier,
       issuer                   Name,
       thisUpdate               UTCTime,
       nextUpdate               UTCTime OPTIONAL,
       revokedCertificates      SEQUENCE OF SEQUENCE {
             userCertificate    CertificateSerialNumber,
             revocationDate           UTCTime,
             crlEntryExtensions Extensions OPTIONAL } OPTIONAL,
                                -- If present, version must be v3
crlExtensions                   [0] Extensions OPTIONAL }}
```

ISO/IEC and ANSI are also developing a set of standard extensions for use in the extensions fields of CRLs. The extensions currently proposed fall into two categories. These categories and the specific extensions proposed are as follows:

a) **Basic CRL Extensions:** These CRL and CRL entry extensions allow a CRL to include indications of revocation reason, to provide for temporary suspension of a certificate, and to include CRL-issue sequence numbers to allow certificate users to detect missing CRLs in the sequence from one CA.

i) *CRL Number*: This CRL extension field conveys a monotonically increasing sequence number for each CRL issued by a given CA. This field is not considered essential but may prove useful for the PKI, e.g. for simplifying management software which may wish to track all CRLs serially.

ii) *Reason Code*: This CRL entry extension field identifies a reason for the certificate revocation. This field is important for the PKI because it will permit the issuing of different CRLs for routine revocations and compromise revocations. Dependent upon application environment and policy, it may be acceptable for some applications to only check the list of compromise revocations. Furthermore, some applications may be able to make direct use of the revocation reason by feeding it back to the user or, potentially, reacting differently to different reasons.

iii) *Expiration Date*: This CRL entry extension field indicates the expiration date of a hold entry in a CRL. It is required for a "hold" reason code and inapplicable otherwise. No PKI requirement for this feature has yet been identified.

iv) *Instruction Code*: This CRL entry extension field provides for inclusion of a registered instruction identifier to indicate the action to be taken on encountering a held certificate. No PKI requirement for this feature has yet been identified.

v) *Invalidity Date*: This CRL entry extension field indicates the date at which it is known or suspected that the ptivate key was compromised or that the certificate should otherwise be considered invalid. No PKI requirement for this feature has yet been identified.

b) **CRL Distribution Points and Delta-CRLs:** These certificate and CRL extensions allow the complete set of revocation information from one CA to be partitioned into separate CRLs, and support the use of partial CRLs indicating only changes since the preceding CRL issue. Such means of limiting the size of CRLs are necessary because large CRLs could result in excessive network traffic and excessive processing overheads in certificate-using end entities.

i) *CRL Distribution Points*: This certificate extension field identifies the CRL Distribution Point(s) to which a certificate-using end entity should refer to ascertain if the certificate has been revoked. One distribution point is identified to contain certificate revocation indications, regardless of revocation reason. Optionally, one or more additional distribution point may be identified as

containing revocation entries for a limited set of revocation reasons. A certificate-using end entity should obtain and use the CRL from the applicable distribution point, unless it can obtain a current complete CRL from the CA itself. This extension is non-critical. If a certificate-using end entity does not recognize this field, then that entity should only accept the certificate if it can acquire and check a complete CRL from the CA; that the latter CRL is complete is indicated by the absence of an Issuing Distribution Point extension field in the CRL. This field is important for the PKI as it provides the means for managing CRL sizes.

ii) *Issuing Distribution Point*: This CRL extension field identifies the CRL distribution point for this particular CRL, and indicates if the CRL is limited to revocations for end-entity certificates only, for CA-certificates only, or for a limited set of reasons only. The CRL is signed by the CA's key - CRL Distribution Points do not have their own key pairs. This field is important for the PKI as, in conjunction with the CRL Distribution Points field, it provides the means for managing CRL sizes.

iii) *Delta CRL Indicator*: This CRL extension field identifies a CRL as being a delta-CRL only. A delta-CRL contains only the new CRL updates since the preceding CRL issue. No PKI requirement for this feature has yet been identified.

In addition to the above CRL extensions, both the authority key identifier extension and issuer alternative name extension, described previously as certificate extensions, may also be used as CRL extensions.

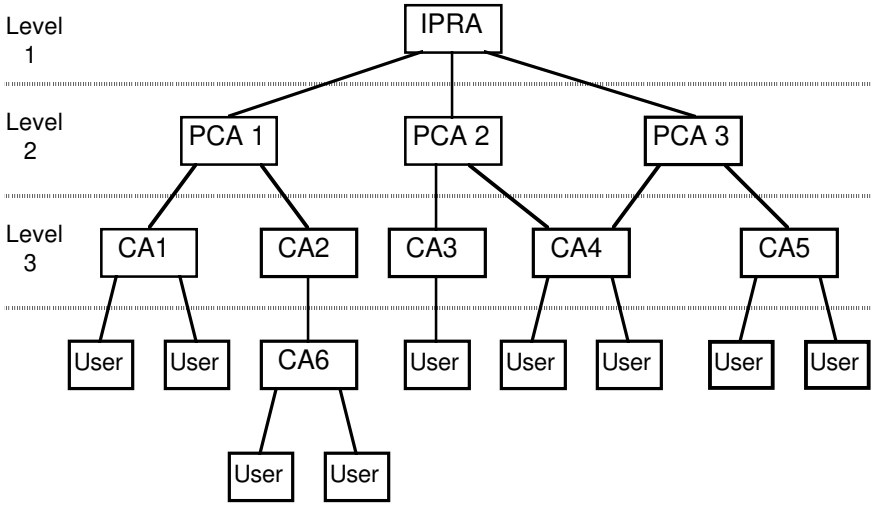
It is considered important for the PKI to employ the two extensions relating to CRL Distribution Points, in order to be able to control the sizes of CRLs without artificially shortening all certificate lifetimes. Therefore, the v3 format for certificates and v2 format for CRLs need to be employed. Using these extensions, each certification authority will partition its CRL into multiple CRLs, each associated with one CRL Distribution Point. The number of currently-valid certificates per CRL distribution point should be limited to some fixed value. The community of end entities associated with one CRL distribution point might be selected along communities of interest or geographical lines, in order to optimize directory access efficiency. These extensions also support the provision of a separate CRL to notify of revocations resulting from key compromises, therefore provide the means of implementing a comparable mechanism to the MISSI Compromised Key List (CKL) mechanism.

4.1.7 Introduction to Certificate Chains

Certificate chains are required because a public-key user is only initialized with a limited number (often one) of assured certification authority public keys.

In general, processing of a certificate chain needs to consider the trust associated with each certificate. If one homogeneous security policy applies to all certificates, this is not an issue. However, when multiple security policies are involved (e.g., when policies vary in different federal departments or agencies or when interoperation with external organizations occurs) certificate chain processing becomes a complex subject.

The best early groundwork on application of public-key infrastructures in multiple-policy environments was done in the Internet Privacy Enhanced Mail (PEM) project [BAL1, KAL1, KEN1, LIN1]. This work constituted a primary input to the MITRE PKI Study [NIS1]. The PEM model defines a hierarchical certification authority structure, as illustrated below.



PEM Certification Authority Structure

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.

(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.

PEM 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 DIT) to the name of the CA itself. The trust associated with a PEM certificate chain is basically 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 PEM certification authority hierarchy model is a valuable starting point in the development of general multiple-policy certification authority structures. However, as a general model for use beyond the PEM environment, it has several deficiencies, including:

(a) The pure top-down hierarchy, with all verification of chains starting from the root, is too restrictive. It must be possible to allow verification of certificate chains starting with a certificate from a user's own domain, rather than mandating verification commence at the top of a hierarchy. The reason is that the former domain is often the most trusted, and because initialization and key-pair-update operations are best conducted between an end entity and a local management system. Policy controls need to be implementable in or near the certificate user's domain. With interoperating domains, it is not feasible to establish one authority who is trusted by all domains for all policies in use. Cross-certification must be possible.

(b) The name subordination rule introduces undesirable constraints upon the X.500 naming system an organization may use. (See 4.3 for further discussion.)

(c) Use of the PCA concept requires knowledge of individual PCAs to be built into certificate chain verification logic. In the particular case of Internet mail, this is not a major problem — the PCA name can always be displayed to the human user who can make a decision as to what trust to imply from a particular chain. However, in many commercial applications, such as electronic commerce or EDI, operator intervention to make policy decisions is impractical. The process needs

to be automated to a higher degree. In fact, the full process of certificate chain processing needs to be implementable in trusted software.

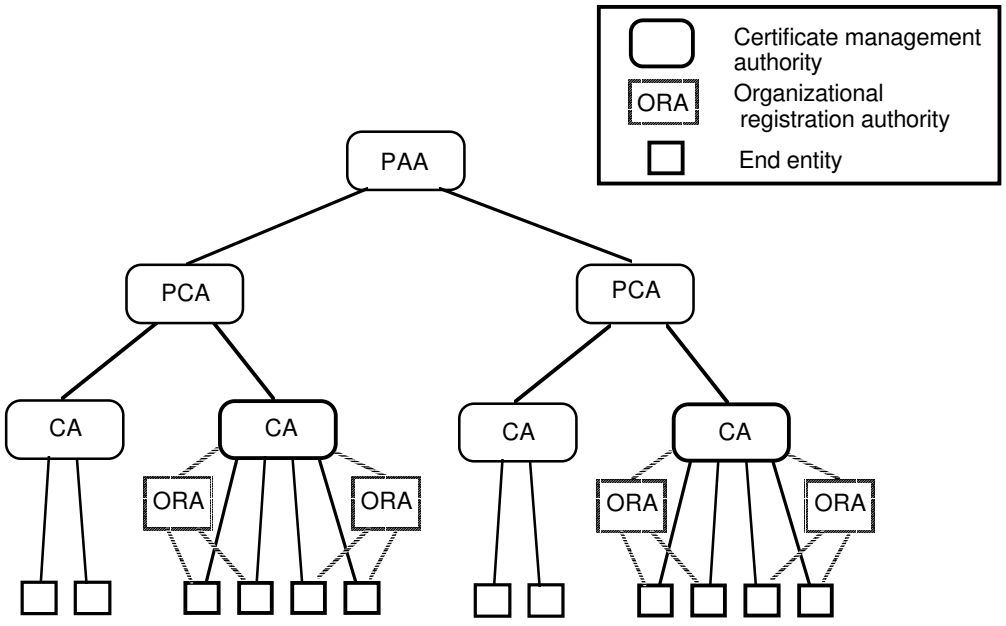
Because of the above shortcomings, it is recommended that a more flexible architecture than the PEM one be adopted for the PKI. The following subsection presents some available options making use of the X.509 v3 certificate format and the standard certificate extensions.

4.2 Infrastructure Architecture

Early federal PKI planning was based on the architecture recommended in the 1994 PKI Study [NIS1], which assumed use of X.509 v1 or v2 certificates. While that architecture recommendation was sound, given the v1/v2 certificate format assumption, the emergence of the v3 certificate format opens up several new architecture alternatives. This report recommends potential relaxation of some of the constraints inherent in the PKI Study model to allow further architectural flexibility, given use of the v3 format.

4.2.1 Early PKI Model

The following diagram illustrates the PKI architecture derived from the PKI Study recommendations, in terms of certificate-issuing relationships:



PKI Study Model Architecture

The primary architectural components are:

(a) *Policy Approval Authority (PAA)*: An authority which establishes the overall infrastructure security policy and creates guidelines that all subordinate entities must follow. The PAA also acts as a "root" certification authority, issuing certificates for the next tier of certification authorities (PCAs).

(b) *Policy Certification Authority (PCA)*: An authority which establishes policy for a single organization or single community of interest. A PCA also acts as a certification authority for the next tier of certification authorities (CAs).

(c) *Certificate Authority (CA)*: In this model, this term denotes certification authorities other than the PAA or PCAs.

(d) *Organizational Registration Authority (ORA)*[7]: An ORA helps an end entity which is physically far from that end entity's CA in registering with that CA and obtaining a public key certificate. An ORA does not include certification authority functions.

(e) *Certificate Management Authority* (CMA): This is a general term which embraces all certification authority types, including PAA and PCA.

This report also introduces the following architectural terms:

(f) *Certificate Management Node*: A term covering all of the infrastructural components associated with a CMA, including components to support ancillary functions (e.g., archival, secure time-stamping) as well as basic certificate issuing and management functions. A certificate management node administers a specific set of digital signature entities, encryption entities, and/or subordinate certificate management nodes. Typically a management node will contain a single physical certification authority, but it might have multiple physical certification authorities, with distinct names and key pairs (e.g., to support a particularly large user community, or to support multiple technologies or algorithms).

(g) *End entity*: An entity which is either a digital signature entity, encryption entity, or both[8]. See introduction to Section 3.

---

[7] The term *Local Registration Agent* used in the ANSI X9.30 and X9.31 standardization work is considered equivalent to the term *Organizational Registration Authority* used here.

[8] One piece of physical equipment may perform both roles of encryption entity and digital signature entity. When this occurs, some efficiencies can be gained by combining PKI operations pertaining to both functions. Note, however, that interdependencies may result, e.g., it is generally recommended that digital signature be applied to a message prior to encryption.

The above model is based on the PEM model (which also assumes v1 or v2 certificates), and shares the following PEM limitations (see 4.1.7 for more detail):

(a) All certificate chains start from the root, i.e., the PAA certification authority; verification of certificate chains cannot start with a certificate from a user's own domain; the PAA must be trusted by all domains for all policies in use.

(b) Cross-certification is not possible (although the possibility of the PAA cross-certifying external structures has been under study).

(b) A name subordination rule is required. (See 4.3 for further discussion.)

(c) The PCA concept is a highly restrictive means of associating policies with certificate chains.

4.2.2 X.509 Version 3 Capabilities

The policy and constraint extensions in X.509 v3 were designed to accommodate the following specific requirements:

(a) It should be possible for certificate chains to start in the local security domain of the public-key user system. There should not be a need for the entire infrastructure to depend upon "root" certification authorities through which all chains must pass.

(b) Any security domain should be able to stipulate which certification authorities in other domains it trusts and for which purposes.

(c) Chaining through multiple policy domains (e.g., government, commercial, and private inftastructures) should be supported.

(d) Certificate chain processing needs to be automatable and self-contained. This is necessary to permit trusted hardware or software modules to be implemented which perform the certificate chain processing functions.

(e) Certificate chain processing should not need to depend upon local user interactions in end entity systems.

(f) Certificate chain processing should not need to depend upon the use of trusted local databases in end entity systems.

(g) There should be provision for regular key-pair updating, and additional key-pair updating under exceptional conditions, for all certification authorities.

(h) Naming should not be unnecessarily constrained, i.e., X.500 name structures considered natural for organizations or geographical areas should not need adjustment in order to accommodate certification authority requirements.

The above requirements cannot be satisfied by the PEM certification authority structure nor any other structure built using the v1 or v2 X.509 certificate format.

The new capabilities offered through the X.509 v3 policy and constraint extensions greatly facilitate interoperation between the federal PKI and external infrastructures. They also make it possible to relax some of the internal architectural restrictions inherent in the model outlined in 4.2.1; such possibilities are discussed in the following subsection.

4.2.3 PKI Model Variations Using X.509 Version 3

Making use of X.509 v3 extensions, it would be possible to relax the PKI model in the following ways:

(a) Remove the restriction that the only certification authorities which can associate policy information with a certificate chain are those certification authorities at the second tier of the hierarchy, i.e., the PCAs. The X.509 v3 certificate policies extension allows any certification authority to associate policy information with a certification chain, and there is no advantage in having the PKI restrict this ability. This does not prevent the PKI having a general guideline that the second-tier certification authorities always associate policy with a certificate (i.e., always include a certificate policies extension in their certificates). Nevertheless, the result of this change is that the term *Policy Certification Authority* loses its significance. (Any certification authority can be a PCA.)

(b) Remove the restriction that certificate chains must start at the root of the hierarchy. Even if all chains were required to pass through the root of the hierarchy, certificate chains may start with a public key of a certification authority more closely associated with a verifing or encrypting end entity (e.g., the "local" certification authority to that end entity).

(c) Extending (b) further, remove the requirement that all certificate chains which are internal to the federal PKI pass through a certification authority at the root of the hierarchy. This would permit certificate chains to be internal to sub-domains of the federal PKI, removing the need for the root to be trusted for all policies and also potentially leading to performance improvements.

(d) Permit cross-certification internal to the federal PKI, in circumstances where the root certification authority is not necessarily trusted for the policy in use, or where performance benefits are possible (due to reduced chain length).

(e) Remove the restriction of only three hierarchical certification authority tiers to the federal PKI.

It is recommended that serious consideration be given to adopting the above relaxations.

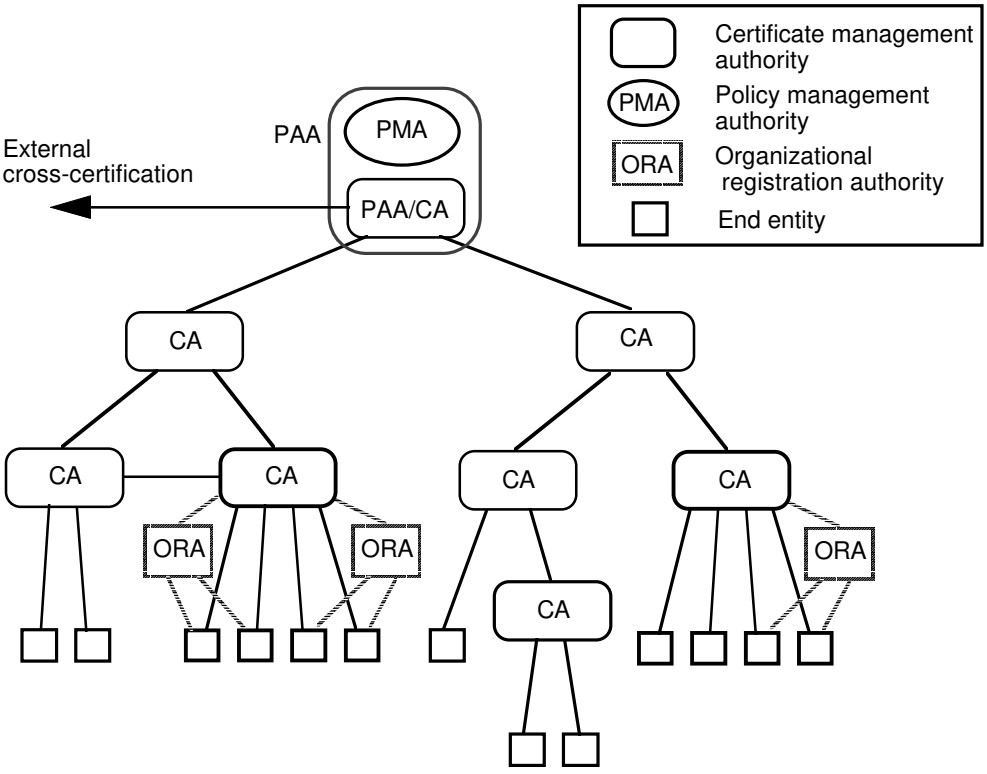
Note that if (a) is adopted, there are now potentially many more policies to manage, including both internal and external policies. Consequently, it is recommended that a new function called *Policy Management Authority* (PMA) be recognized. This function will register and coordinate policy definitions.

Other relaxations which would also be possible, but which are not necessarily recommended, especially in early stages of deployment, are:

(a) Relax the hierarchical structure even further, e.g., permit multiple interoperating smaller hierarchical structures. There is no technical reason precluding this, however the PAA-based hierarchical structure is beneficial as a way of controlling delegation of authority (even if certificate chains do not rigidly reflect that hierarchical structure).

(b) Permit cross-certification from arbitrary certification authorities in the federal PKI to external infrastructures. This is not recommended, at least in early stages of PKI deployment, as generation of external cross-certificates requires substantial due diligence and tight controls, therefore is better centralized. It is reasonable to require that only the PAA certification authority be able to issue certificates for external certification authorities.

If the recommended relaxations on restrictions are made, the PKI model can be better illustrated as follows:



Flexible Architecture

Note that, in the above, the PAA concept has been retained but that it now comprises two components — the PMA and the PAA/CA. The PAA/CA is a certification authority, the main significance of which is that it is the only point within the federal PKI which certifies external certification authorities.

With the v3 certificate format, it is proposed that the PKI employ the following standard extensions relating to policies and constraints in the following ways:

(a) *certificate policies*: Certificate management nodes, through coordination with the PMA, will define certificate policies and policy elements appropriate for their subordinate subdomains when suitable preexisting policies or policy elements do not exist. Node certification authorities will include the appropriate policy identifiers in the certificates they issue.

(b) *policy mappings*: Policy mappings, if required, should be managed by the PMA. This authority will specify policy mappings between federal government policies and policies in external infrastructures. Other certificate management nodes may define policy mappings between their own defined policies and other PKI policies.

(c) *basic constraints and policy constraints*: The PMA will specify constraints to apply to certification authorities in external infrastructures. Other certificate management nodes may define name and policy constraints to apply to certification authorities of other certificate management nodes.

In addition, the following general structuring guidelines are suggested for the PKI:

(a) A public-key user entity (digital signature verifier or encryption entity) should be able to verify any public key which its own certificate management node intends it to be able to verify, after being initialized with only a single certification authority public key.

(b) The single certification authority public key referred to in (a) should be the public key of a certification authority in a superior certificate management node. One option which should specifically be supported is the public-key of the immediately superior certificate management node, as this facilitates end-entity initialization and updating of both user and certification authority key pairs.

(c) Every PKI certification authority should sign and make available via publicly-accessible directory services, certificates for all of its immediately subordinate certification authorities.

(d) Every PKI certification authority should sign, and make available via a Directory service accessible by subordinate public-key users, certificates for the public keys of certification authorities in its immediately superior certificate management node.

(e) The PAA/CA certification authority should sign, and make available via a Directory service accessible by all PKI systems, certificates for the public keys of all of its immediately-subordinate certification authotities, all cross-certified external certification authotities, plus other heavily used internal certification authorities. Through chaining, this will give access to all PKI certification authorities and to recognized external certification authorities.

(f) A PKI certification authority may, at its option, sign, and make available via a Directory service accessible by subordinate public-key users, certificates for other PKI certification authorities whose certificates are heavily used by those users.[9]

[9] Note that such certificates generally violate the PEM name subordination rule (see 4.2.7) so may be unusable in certificate chains processed by PEM implementations or other systems outside the federal PKI environment. These certificates are usable within the PKI environment provided the certificate-using system is programmed to recognize and trust all certificates issued by its superior PKI nodes.

The above approach will satisfy all certificate chain requirements for interactions within the U.S. government. If external infrastructures adopt the v3 certificate format and certificate extensions for chain processing controls, then all requirements with respect to external interoperation will also be met. However, if an external infrastructure adopts more restrictive structures, e.g., if it retains v1/v2 certificate formats, then external interoperation may be limited. Therefore, it is recommended that the U.S. government actively promote, in standards forums and with allied governments, adoption of the v3 certificate format and the standard extensions, with the goal of ensuring that different government and private infrastructures will interoperate with full flexibility.

4.3 Naming

The operation of the automated key and certificate management system will depend upon unique X.500 names for all:

(a) digital signature entities[10];

(b) encryption entities; and

(c) certification authorities.

In addition, unique names are required for:

(d) operations personnel involved in the operation of certificate management nodes.

While (d) entities do not necessarily require X.500 naming, and do not need to be distinguished within the name space used for (a), (b), and (c), it is recommended that X.500 naming also be used for these names. This will facilitate such procedures as use of digital signatures for sealing of operations logs by operations personnel. The remainder of this subsection focuses on naming issues with respect to (a), (b), and (c)

The X.500 naming conventions adopted for the PKI should be consistent with other government-wide X.500 schema developments.

Assignment of names may need to take into account the requirement for interoperability with public-key infrastructures of organizations outside the U.S. federal government. Both Internet PEM, and early U.S. government infrastructure proposals based on PEM, place constraints on the names that can be used in interoperating infrastructures.

[10] If one piece of equipment performs both encryption and digital signature functions, the one name may serve the purpose of encryption entity name and digital signature entity name.

The PEM name subordination rule [KEN1], requires that a CA only issue certificates for entities whose names are subordinate (in the X.500 DIT) to the name of the CA itself. This rule is included for a sound reason, namely, to provide a simple automatable means of determining if a particular CA (below the PCA level) is authorized to sign a certificate for a given subject. However, the particular restriction chosen is considered unacceptable for large enterprise environments (including the U.S. federal government) because:

(a) It means that an organization is forced either to distort its overall X.500 naming structure to accommodate artificial DIT nodes for certification authorities or to have organization or organizational units share X.500 names with their certification authorities. The problem with the latter option is that they must then share the one directory entry — this is undesirable because not all directory attributes for the two types of entity are the same. For example, it is desirable to be able to list different directory attributes (e.g., telephone number, access controls) for the NASA organization and for the certification authority operated by NASA.

(b) It would force a restriction that only one certification authority (and one certification key pair) can exist at any one organization or organizational unit node. However, there are various practical reasons for wanting more than one certification authority at such a node (e.g., certification authorities with different assurance levels for different categories of user, standby certification authorities, and certification authorities of different technologies).

These problems have been overcome in the development of the X.509 v3 certificate. The Name Constraints extension has introduced the "subordinate-to-CAs-superior" rule as an alternative to PEM's "subordinate-to-CA" rule. (The former specifies that an entity's distinguished name must have as a prefix that of the DIT entry which is the immediate superior of the entity's CA entry; the latter specifies that an entity's distinguished name must have as prefix the distinguished name of the entity's CA.)

As an alternative to the Name Constraints extension, the subtrees constraint in the Basic Constraints extension also achieves similar goals, although it does not precisely reflect a name subordination rule.

It can be anticipated that the Internet community will modify their name subordination rule when certificate management is updated to use the X.509 v3 certificate. The updated model will likely use either the subtrees constraint or the "subordinate-to-CAs-superior" rule.

4.3.1 Naming of Digital Signature Entities

A digital signature entity is a signer, a verifier, or both. Names for signers are most critical, because they are used in certificates and because they must be recognizable to verifiers (potentially both within and outside the U.S. federal government). A signer needs to possess a private key which ensures the legitimacy of the source of a digital signature; a corresponding public key is distributed to verifiers in a certificate which binds the public key to the signer's name. Each signer therefore needs a unique X.500 name.

A verifier does not necessarily need an X.500 name. Nevertheless, a verifier needs to be registered with the PKI because it is necessary to install key material which enables a verifier to determine the integrity of a signer's public key. It is recommended that X.500 naming be used for verifier names. This will facilitate management of information regarding people performing multiple roles, e.g., signer and verifier.

As pointed out in the 4.3 introduction, there is a potential need for a signer name to bear a special relationship to the name of the certification authority that issues its certificate. Signer names should, at least, follow the "subordinate-to-CAs-superior" rule in relationship with their certification authority's name. For compatibility with external infrastructures which are slow to adopt the "subordinate-to-CAs-superior" rule, it may be necessary to conform to PEM's "subordinate-to-CA" rule for an interim period.

4.3.2 Naming of Encryption Entities

Encryption entities must have unique names to support the identification and authentication requirements of key management. Given the requirements for public-key based key distribution and for standard protocols, the key management system should be planned to operate with X.509-based public key certificate infrastructures and X.500-based naming.

A name for an encryption entity could be either:

(a) an identifier of a physical piece of encryption/decryption equipment; or

(b) an identifier of a person associated with an encryption/decryption process.

Case (b) applies when the encryption/decryption functionality resides in a piece of equipment but is only enabled when a particular person is actively using that equipment. For example, the equipment might be a sharable workstation or PC, but it assumes an encryption/decryption identity when a particular person logs on to the equipment (e.g., authenticates with a password or inserts a smart card). In this case, the key material is associated more with the person than with the equipment.

As pointed out in the 4.3 introduction, there is a potential need for an encryption entity name to bear a special relationship to the name of the certification authority that issues its certificate. Encryption entity names should, at least, follow the "subordinate-to-CAs-superior" rule in relationship with their certification authority's name. For compatibility with external infrastructures which are slow to adopt the "subordinate-to-CAs-superior" rule, it may be necessary to conform to PEM's "subordinate-to-CA" rule for an interim period.

4.3.3 Naming of Certification Authorities

Certification authority naming may be impacted by the need to interoperate with external public key infrastructures such as the PEM infrastructure or the prototype MISSI infrastructure. Such interoperation means that systems supported by one infrastructure must be able to use and verify certificates issued in the other infrastructure. For example, any certification authority operated by the U.S. federal government could be an Internet CA and be certified, for external interoperation purposes, by one or more Internet PCAs. (The same certification authority will also typically be used for fully government-internal digital signature purposes.)

Using the "subordinate-to-CAs-superior" rule, it will be possible for an organization or organizational unit to have one or more subordinate CAs, any of which can issue certificates for any entity subordinate to the organization or organizational unit. As an illustration of this approach, assume that a certification authority associated with a particular organization or organizational unit DIT node is indicated by use of a relative distinguished name of the form "OU=ca name" immediately subordinate to that node. For example, the X.500 node "C=US, O=Gov, OU=DoD" might have an associated certification authority with name "C=US, O=Gov, OU=DoD, OU=DoD CA No. 1". The certification authority can only issue certificates for entities whose X.500 names are subordinate to that of the superior node, i.e., names starting with "C=US, O=Gov, OU=DoD, ...".

Using the "subordinate-to-CAs-superior" rule, there can be multiple CAs associated with one DIT node, with any of these CAs being able to generate certificates for entities with names subordinate to that DIT node. However, it is not possible for any CA to issue certificates for end entities with names outside the span of its associated DIT node.

4.4 Basic Key Management Functions

This subsection addresses the following basic key management functions:

(a) encryption key generation;

(b) encryption key distribution;

(c) public-private key pair generation; and

(d) end-entity initialization.

Functions associated with key archival, recovery, and certificate management are addressed separately in other subsections.

4.4.1 Data Encryption Key Generation

Data encryption keys will generally be generated at the lowest practical PKI level. This generally implies generation within one of the encrypting/decrypting systems or via on-line agreement between a pair of such systems. For a particular instance of encryption, the data encryption key is generated and distributed in either of the following ways:

(a) *Key transfer*: The encrypting system generates, via a random or pseudo-random process, an encryption key. The decryption key (which, assuming a symmetric cryptosystem, is the same as the encryption key) is conveyed to authorized decrypting systems via a key distribution mechanism which protects it during communication, e.g., the symmetric key is protected by encrypting it under an RSA public key of the decrypting entity to which it is sent.

(b) *Key agreement*: The encrypting and decrypting systems establish an encryption/decryption key via an on-line key derivation exchange (such as Diffie-Hellman key derivation [DIF1, FOR1] or MISSI's KEA).

In both of cases (a) and (b), the quality of the key generated (in terms of its resistance to cryptanalysis) depends upon the randomness characteristics of a value input to the key generation process. Ideally, a truly random process should be used in key generation, but such a process is rarely available in equipment used for protection of unclassified information. More commonly, a pseudo-random process is used, operating upon a secret, unpredictable input *seed* value (typically an output from the previous cycle of the same process). A pseudo-random process with appropriate characteristics must be used. However, the strength of the entire process still depends upon a suitably-unpredictable initial *seed key*. Secure distribution of seed keys should be an optional PKI function provided in conjunction with end-entity initialization.

4.4.2 Encryption Key Distribution

Encryption keys are generated as discussed in 4.4.1. Distribution of symmetric encryption keys can employ any of a variety of methods such as:

(a) on-line key agreement, in which session key generation and distribution both occur as part of one on-line exchange;

(b) on-line key transfer, in which a session key is transferred between two encryption entities, either as part of their basic communication protocol or in a higher-layer key management protocol;

(c) store-and-forward key transfer, in which protected versions of the decryption key are attached to an encrypted message (typically multiple copies of the decryption key are attached, each protected under a long-term key relationship with a particular recipient); commonly the protection process involves encrypting the decryption key under an RSA public key of each recipient, but systems using the Diffie-Hellman technique with long-term key pairs can also be used (a public Diffie-Hellman component is distributed in a certificate as a form of public key).

Regardless of the key distribution method, the parties to the key distribution process generally need to identify and authenticate each other as part of that process. The PKI should be capable of supporting these identification and authentication needs of encryption key distribution. This involves maintaining a capability for public-key certificate generation and distribution. This capability is discussed in 4.2.

The requirement may also arise for distribution of encryption keys via trusted key servers. This approach is useful when information items are distributed by store-and-forward means using untrusted facilities, such as public file servers, bulletin boards, or electronic mail. The approach enables an encryptor of information to specify the set of authorized decryptors in terms of a group-based, role-based, or multi-level access control policy. A trusted server has the ability to authenticate systems requesting decryption keys and to make decisions as to whether or not the requesting system is authorized to be sent the key. Such a capability is described in [FOR2]. This capability might be considered by the federal government as a possible optional PKI feature.

4.4.3 Public-Private Key-Pair Generation

Any entity acting as an encryption entity or digital signature signer supported by the PKI needs to have a public-private key pair. This key pair can be generated in either of the following ways:

(a) The key pair is generated within the encryption entity or digital signature entity which is to house the private key. The public key is transferred, with high integrity, to the PKI which then generates a certificate for that key for the purposes of further distribution.

(b) The key pair is generated by the PKI. The private key is transferred, with high confidentiality and integrity, to the encryption entity or digital signature entity which is to house that private key. The PKI retains a copy of the public key, in order to generate a certificate for that key for distribution to verifiers.

As noted in 3.1, method (a) is recommended for digital signature entities, because the secrecy of the private key is particularly critical if non-repudiation properties are to apply. Furthermore, there is no need for any entity other than the signer to maintain a backup or archival copy of that private key (if the private key is lost, a new one is simply created). For encryption entities, both method (a) and method (b) should be supported.

Regardless of which system generates the key pair, the quality of the key pair generated (in terms of its resistance to cryptanalysis) depends upon the randomness characteristics of a value input to the key generation process. Ideally, a truly random process should be used in key generation, but such a process is rarely available in equipment used for protection of unclassified information. More commonly, a pseudo-random process is used, operating upon a secret, unpredictable input *seed* value (typically an output from the previous cycle of the same process). A pseudo-random process with appropriate characteristics must be used. However, the strength of the entire process still depends upon a suitably-unpredictable initial *seed key*. Secure distribution of seed keys should be an available PKI function provided in conjunction with end-entity initialization.

4.4.4 End-entity Initialization

End-entity initialization involves establishing, in a digital signature entity or encryption entity, initial key material to enable that entity to function thereafter with only on-line interactions with a PKI node. In the same process, any private keys generated by the end entity but requiring to be backed up or archived by the PKI are transferred to the PKI node. The initial key material installed in the end entity may include:

(a) unique identification information for the entity;

(b) private key information used to support encryption key agreement or transfer, and/or private key information used to support authentication for encryption key distribution purposes;

(c) private key information, used to support digital signature generation;

(d) public keys of one or more certification authorities, to enable other certified public keys to later be verified;

(e) seed keys for use in key generation (see 4.4.1 and 4.4.3).

In the case of digital signature entities or encryption entities based on personal tokens (e.g., smart card or PCMCIA card), end-entity initialization equates to initialization of the token. In this case, the initialization involves either:

(a) the token internally generating its own key materials and emitting public key component(s) for PKI certification and, in some cases, private key component(s) for PKI backup/archive; and/or

(b) the token being loaded with initial key material generated by the PKI.

In other cases (including software implementations of client cryptographic functions), the end-entity initialization process may be partially conducted on-line. It cannot be conducted purely on-line, because the entity could not adequately authenticate itself to the PKI node. The process must involve a manual step between the end entity and either a PKI node or an ORA, such as:

(a) transfer of initial key material via data media, e.g., a data key or diskette, handled in accordance with special manual controls;

(b) transfer of initial key material via an intelligent device, such as a smart card or data transfer device (DTD), with password(s) or PIN(s) being required to enable or unlock the device;

(c) manual entry of an initial password, communicated from a certificate management node to an on-site operator via means such as secure telephone or secure mail.

In all cases, (especially case (c)), the initial manually assisted exchange does not necessarily transfer all required initial key material. However, the data transferred is sufficient to authenticate the establishment of a secure on-line session with a PKI node to enable remaining initial key material to be transferred on-line.

There is no standard on-line protocol for initializing cryptographic devices and there is no broad-based standardization activity to develop such a standard. Therefore, the PKI may need to support a variety of end-entity initialization methods and a variety of end-entity initialization on-line protocols.

Entity-specific initialization of both software-only and token-based end-entities (3.5 and 3.6) must be considered, in addition to domain-specific initialization of the end-entity software application (e.g., communicating to an end-entity policy parameters defined by that entity's management node). In the case of token-based end-entities, use of a common (low-level) cryptographic API between the token and the end-entity application is recommended, to allow a given end-user application to support multiple token types. Regarding the interworking of an end-entity application with the certificate management node, this may be achieved by either:

(a) building such applications on top of a non-proprietary (high-level) cryptographic API, with functions calls through this API supported by a toolkit compatible with, and capable of communicating with (via either proprietary or non-proprietary protocols), a PKI management node; or

(b) designing such applications to interface directly with certificate management nodes, through use of a standardized protocol specification.

4.4.5 Public-Private Key-Pair Updating

All public-private key pairs need to be updated periodically; this applies to key pairs used for digital signature purposes or for key agreement, key transfer, or authentication in conjunction with encryption. The purposes of updating are to restrict windows for cryptanalytic attacks and to limit the exposure of a key compromise to a well-defined period. Depending upon the technology employed and the application environment, the update period may be comparatively long (weeks, months, or, possibly, years).

The validity period of a certificate indicates to a certificate user (i.e., a public key user) the period for which both of the following can be assumed: (a) the public key is valid, and (b) the binding between the public key and other information in the certificate (notably, the distinguished name) is valid. A certificate validity period in a certificate can be overridden by a revocation. The validity period for using a private key corresponding to a public key may be quite different to the certificate validity period for the corresponding public key. The situations with key pairs for digital signature purposes and key pairs for encryption-key-establishment purposes are discussed separately below.

With digital signature key pairs, in order to limit exposure in the event of private key compromise, the validity periods of private keys are restricted to well-defined intervals. Key-pair updating means that an existing private key is no longer used (and can be destroyed) and a new key pair is established; the private key from the new key pair is used for subsequent signatures. However, when a new key pair is established, the old public key still needs to be used to verify signatures. Hence, the public key validity, and the corresponding certificate validity, will generally need to extend somewhat beyond private key validity. In fact, the public key validity (and the validity of some certificate for that public key) may well need to extend many years after the private key is no longer

used. When signing times can be trusted, it is valuable for a verifier to know the valid lifetime of the signing (private) key. An additional field has been included in a v3 certificate extension for this purpose.

CA signature key pairs are one case of digital signature key pairs as discussed above. However, it is important to note that the valid lifetime of a public-key certificate is limited not only by its stated validity period but also by the validity of the affixed CA signature. Therefore, a CA should ensure that the validity period of its own public key (and some corresponding certificate) extends over the intended validity period of any certificate it is signing.

For any signing party, especially any CA, multiple public keys may be valid for signature verification purposes at any time. Hence, multiple valid live certificates for that signing party may exist at any time. When implementing a CA, it is necessary to provide a means for a certificate verifier to be able to find the appropriate CA certificate. No suitable means is provided in the X.509 v1 or v2 certificate definitions . However, the v3 certificate extensions provide for a CA key identifier for this purpose.

With a key pair used for encryption key establishment (e.g., KEA or RSA key transfer), public-key validity (and corresponding certificate validity) are generally shorter than private key validity. This is because the public key is used for encryption and the private key for decryption, and decryption typically still needs to be possible after encryption using the same key pair has ceased. Key pairs are updated periodically, with the update period corresponding to the public-key validity period. Public-key certificate validity should also be made the same value (although it is possible to make certificate validity shorter, recognizing that multiple certificates will then be required to cover the public-key lifetime). Private key validity is not reflected in certificates. However, the private key holder will generally recognize a validity period for that key (based on assessed resistance to cryptanalysis). After that time, information encrypted on the basis of that key (e.g., a KEA private key) is no longer considered secure.

As part of the key update process, any private keys generated by the end entity but requiring to be backed up or archived by the PKI are transferred to the PKI.

The key-pair updating process generally involves either:

(a) a repeat of the end-entity initialization process; or

(b) an abbreviated process in which the manual intervention step is not needed; data from the previous key cycle (e.g., a private key) is used to secure an on-line transaction which accomplishes the requisite updating of key material.

There is no standard protocol for key-pair updating and there is no broad-based standardization activity to develop such a standard. Therefore, the PKI will most likely need to support a variety of on-line protocols for this purpose.

Updating of the key pair for a certification authority requires special consideration, as follows:

(a) There must be an overlap period in which a new key pair and old key pair are both considered valid. Other certification authorities which certify this authority's public key must maintain valid certificates for both public keys covering the overlap period.

(b) When a new key pair is generated, new certificates must be issued for all public keys of subordinate entities which have a lifetime extending beyond the overlap period. The existence of the new certificates must be notified to the subordinate entities concerned.

(c) The new certification authority public key must be transferred with high integrity to those subordinate entities that depend upon knowledge of that key to initiate processing of a certificate chain.

(d) The new certification authority public key must be transferred with high integrity to all certification authorities, both within or external to the PKI, that issue CA-certificates for the certification authority that has updated its key pair.

(e) During the overlap period, dual certificate revocation lists (see 4.2.5), signed by each of the active keys, should be posted.

All aspects of key pair updating should be dealt with automatically by the products involved and be transparent to the end user.

4.5 Key, Certificate, and CRL Backup/Archival

Keys and certificates may require backup, to support short-term recovery scenarios, and/or archival, to support long-term evidence retrieval. The requirements and services vary in the cases of encryption and digital signature, and are described in the following subsections.

In general, backup facilities should be provided at every certificate management node. Provision should also be made for on-line transfer of data for longer-term backup or archival at a node's superior node. Given such provision, it is possible to have nodes which do not perform their own backup. One or more central archive facilities should be provided as part of the PKI, for backing up highly sensitive data and for long-term

archival. All data requiring archiving for lengthy periods (e.g., exceeding three years) should be archived at a central facility.

A signed data item archived for a very long period may outgrow its own expiry date, and require (periodic) resigning by currently trusted keys. Such resigning will be a local function of an archival node and will be governed by local policy.

4.5.1 Key Backup/Archival for Confidentiality Services

When sensitive information is stored in an encrypted form for a significant period of time, there may be a requirement to be able, subject to appropriate authorization, to recover decryption keys from a reliable backup/archive. This requirement applies to such applications as file encryption and electronic mail, where the only copy of a piece of sensitive information may be an encrypted copy. It does not apply to short-term encryption such as communication session encryption, where a plaintext copy of the encrypted data usually exists in at least one end system. The need to use a plaintext recovery process based on backup/archival can result from, for example:

(a) loss of a decryption key due to equipment or media failure;

(b) loss of a decryption key due to accidental or deliberate actions of a person with control over such key; or

(c) the requirement to override controls on an instance of decryption, e.g., as a result of an interception order from a law enforcement agency.

The PKI should incorporate a capability to backup/archive key material to enable the recovery, under appropriate circumstances, of decryption keys used for such applications as file encryption and electronic mail (as discussed above). This capability can be provided at any certificate management node, applying to the encryption entities subordinate to that node.

If encryption key distribution is based on key transfer or key agreement using public-key techniques with long-term key pairs, and if appropriate end-entity initialization and public-private key pair update procedures are followed (see 4.4.4 and 4.4.5), then backup/archival of the private keys of these key pairs can satisfy the above requirement. (This assumes that encryption keys can then be derived from other stored information, such as header fields stored with encrypted files or messages.) Depending upon the key establishment method, there may or may not be a need for backup or archival of the corresponding public keys or public-key certificates. With RSA key transfer, there is no need to backup/archive public keys. However, with Diffie-Hellman-based key agreement, there is a need to backup/archive public keys in order to be able to recover encryption keys.

Following the above reasoning, end-user long term decryption keys (e.g., RSA key transfer keys) should be archived permanently, with migration of the archived data from a local management node to a central archiving facility as such data ages. Based on a default encryption key (RSA key transfer key) lifetime of 2 years, with a one-year (50%) key overlap period, this requires that for an n-year period, storage for archival of (n+1) decryption key pairs per user is required.

The backup facility used for this purpose must be highly secure. The backed-up data must be stored in an encrypted form. A split-knowledge system should be used to control access to the corresponding decryption capability, e.g., two authorized administrative personnel should be required to independently present passwords in order to initiate recovery of a backed-up key.

For encryption entities subject to this type of backup/archival, the goals of key escrow (as addressed by the Escrowed Encryption Standard [FIPS3]) can also be satisfied. Assuming the PKI nodes are considered sufficiently trusted, any decryption key can be recovered in response to authorized law enforcement or intelligence interception requirements.

4.5.2 Confidentiality Key Recovery

In support of the key backup/archival facility described in 4.5.1, there is a requirement for a capability and associated operational procedures to effect key recovery. Procedures to be specified include:

(a) procedures to request key recovery, for use by a person with operational or administrative responsibility for an encryption entity or by another person authorized to request key recovery;

(b) procedures to authorize a request for key recovery; and

(c) procedures to effect an appropriately-authorized key recovery, making use of split knowledge key materials held by two or more authorized persons (e.g., persons involved with the operation or administration of a certificate management node).

The principles underlying the above procedures will need to be specified as part of the PKI security policy.

If the purpose of a key recovery is to reestablish lost key materials in an encryption entity, the process will generally lead to an end-entity initialization process, or an adaptation of that process tailored for the key-recovery scenario.

4.5.3 Certificate and CRL Archival for Digital Signature Services

With digital signature services, there is no need to backup or archive private keys. If a private key is lost or destroyed, a new private key can be generated for the signer entity. However, there is a need to archive public keys because digital signatures may need to be verified some time after they are generated.

To satisfy non-repudiation requirements, it is furthermore necessary to archive all public-key certificates (including certificates of certification authorities signed by other certification authorities) and a history of certificate revocations, to be able to ascertain if a particular digital signature was valid at the time it was generated.

This archive does not need to be confidential, but it does need high integrity, i.e., there must be high assurance that data in the archive cannot be modified or removed (internal certificate signatures are not necessarily adequate integrity protection because keys may be compromised at later times).

One way to archive public keys with verifiable integrity is to archive the public-key certificate, along with the corresponding certification authority public key (which must itself be stored with guaranteed integrity). The integrity of the certificate here, however, becomes questionable after the expiry of the lifetime of the key that signed it. Other measures for archive integrity are typically relied on, e.g., appropriate technologies and/or procedural techniques for conventional secure storage.

Because the items being archived are in the form of signed certificates, their validity can be easily ascertained by any system at archival time. Hence, archival does not necessarily need to be linked to any particular certificate management node. This supports the recommendation that a central certificate archive system be provided as part of the PKI.

Similar to the archival of certificates, archival of certificate revocation lists (CRLs) is required to aid in the non-repudiation service. Note that while timestamps within CRLs are useful for indicating when the next CRL is expected, they are of use with respect to determining whether a signature occurred before or after a key revocation only if the signature was timestamped in a trusted way or countersigned by a trusted notary. In the absence of such timestamp/notarizing information, the time at which a signature was created cannot be determined (any time indication embedded within signed user data is simply an uncorroborated assertion by the possessor of a possibly compromised signature key).

4.5.4 Digital Signature Certificate Recovery

In support of the certificate archival facility described above, there is a requirement for a capability and associated operational procedures to effect certificate recovery.

Recovery of certificates from an archive which supports digital signature services does not necessarily require special authorization or controls as the archived data is not, in general, confidential. However, procedures will need to be specified for requesting recovery of a certificate (or set of related certificates) and for conveying the results of such recovery, with high integrity, to the requestor.

A node might depend upon archival services of a superior node, in which case requests for retrieval of information would also involve such superior node, and it is recommended that data to be archived for lengthy periods be archived at a central archive facility.

4.5.5 Evidence Retrieval and Interpretation

To verify a digital signature, a verifier carries out standard public-key verification procedures, possibly involving verification of certificate chains as outlined in clause 4.2.7. At a subsequent point in time, it may be required that a signature be re-verified. It is necessary that information required to subsequently re-verify a signature, including possibly that necessary to reconstruct a certificate chain, be available throughout the lifetime of the signature verification key. In particular, such information must be available in the case of signature disputes, i.e., to support the service of non-repudiation. In conjunction with notary services (as discussed in 3.8 and 5.7), this may require retrieval of archived CA public keys, signature public-key certificates, and CRLs from appropriate archives.

If a signature key pair is revoked owing to suspected key compromise, all previous signatures created using that key pair, including those by the authorized user, become open to question (i.e., a successful cryptographic verification of the signature is not adequate to establish validity of the signature). This results because, in the absence of timestamping/oversigning by a trusted notary service or using a trusted hardware time source (e.g., in a cryptographic token), the time of signing cannot be established other than as having occurred at some time after the signature key pair was created. If such timestamping/oversigning has taken place, then the effective time of revocation (which may or may not be the same as that on the relevant CRL, depending on the security policy in place) may be used to distinguish previously valid signatures.

In the case that a signature key pair is revoked before expiration of its normal validity period for reasons other than suspected compromise, revocation should not invalidate all previous signatures. With such a revocation, it is assumed that all copies of the signature private key are effectively rendered unusable (destroyed). It is therefore important for CRLs to convey reason codes indicating reasons for revocation. There is an extension field for this purpose in the v2 CRL format.

The non-repudiation service associated with digital signatures involves a trusted third party resolving a dispute wherein a claimant presents a purported (signature, document)

pair, and claims that the signature is valid. The role of the third party is to carry out an independent signature verification process, and declare the signature either valid or invalid.

Possible reasons for a signature verification by the third party to fail are given below. In the case that no trusted timestamping service was involved, reasons for signature failure include:

(a) The corresponding signature key has been revoked, with reason of compromise. In this case, regardless of when the signature was created (even if by the authorized party prior to revocation), the signature will fail to verify. Such revocation can be established by consultation of the appropriate CRLs, possibly obtained from a CRL archive.

(b) The corresponding signature key is no longer valid. Expired signature keys should be treated similarly to revoked keys (recall that revoked keys are removed from CRLs upon expiry). Once the validity period of a signature key expires, previous signatures which verified successfully will no longer be verifiable; this is necessary since there is no way of differentiating between a signature created within the validity period from one created beyond it. If it is required that the lifetime of signatures be extended, additional measures are required (e.g., through use of by trusted timestamp/notary service).

(c) The corresponding signature is (mathematically) incorrect, e.g., due to modification of information that was signed.

In the case that a trusted timestamping/notary service was involved, additional procedures, as defined by that service, may be used to provide a finer granularity in time resolution.

The process of evidence retrieval and interpretation necessary to support the service of non-repudiation involves the reconstruction of evidence similar to that available to the originally intended verifier at the time of signature creation, but possibly augmented by subsequent changes (including revocation and expiry). This reconstruction is carried out by a trusted third party (e.g., security officer of the central archive facility), who may be required to access appropriate archives as necessary to retrieve information no longer available in public directories, possibly including certificate chains. If, having assembled such information, the verification of the signature, as per the usual signature verification process, succeeds, then the signature is deemed to be valid (decision in favour of verifier); if the signature fails to verify, the signature is deemed to be invalid (decision in favour of the signer who repudiates).

4.6 Audit and Alarm

Every certificate management node will be required to maintain a local audit trail of security-significant events. A subset of these events may need to be reported back to the superior certificate management node. Centralized audit trail accumulation can occur at the central archive facility. A security alarm facility should also be provided at each certificate management node, for signalling such events as suspected local node penetration or privilege violation, or failure of on-line services.

In addition, some end-entity devices may support a capability for uploading audit trail data to a central accumulation point. If so, the PKI will need to support the accumulation and processing of such data.

4.6.1 Local Audit Trail

Every certificate management node will be required to maintain a local audit trail of security-significant events. A large range of events will need to be recorded; following is a partial list:

(a) registration/deregistration of encryption entities, digital signature entities, and certification authorities;

(b) public-private key pair generation;

(c) public-private key pair updating;

(d) end-entity initialization;

(e) certificate generation;

(f) certificate revocation;

(g) confidentiality private key backup/archival;

(h) confidentiality private key recovery;

(i) digital signature certificate archival;

(j) digital signature certificate recovery;

(k) transfer of an encryption entity, digital signature entity, or certificate management node to a new superior node;

(l) registration/deregistration or change in privileges for an individual node operator;

(m) token or smart card loading operation;

(n) node operator logon, password change, or security-significant command;

(o) node operator failed logon attempt or authorization violation;

(p) any abnormal event, e.g., equipment failure, software or hardware self-test failure.

4.6.2 Centralized Audit Data Accumulation

A subset of the local audit trail data accumulated at a certificate management node may need to be reported back to its superior certificate management node. Centralized audit trail accumulation should be available at a central archive facility. The PKI design will need to incorporate a communications protocol for transferring audit trail data (protocol requirements are addressed in 5.2.4).

The particular set of audit trail data to be reported back will need to be determined and reflected in overall PKI security policy and/or local certificate management node security policy.

4.6.3 Security Alarm

Every certificate management node should be required to have an alarm facility, capable of raising an operator alert when certain security-significant events (e.g., software or hardware self-test failure) occur or when thresholds, such as allowable number of node operator failed logon attempts, are exceeded. In general, this alarm facility can be driven by the same data accumulation that feeds the audit trail facilities.

4.6.4 Audit and Alarm Support for End Entities

Some encryption and/or digital signature devices may have a capability to upload audit trail data to a central accumulation point. However, this cannot be assumed to be a standard feature of products of this type, and no standard communication protocol can be assumed (see 5.2.2 and 5.2.4 for further discussion). For those devices that have such a capability, PKI support is desirable but, unless a standard protocol is established, such support would need to be product-specific.

For the purposes of this report, it will be assumed that uploading of end-entity audit trail data will be restricted to data transferred in conjunction with end-entity initialization or key-pair updating.

4.7 Distributed Data Management

Distributed information to be managed includes:

(a) PKI structure information. This includes information regarding which certificate management nodes, encryption entities, and digital signature entities are administered by each certificate management node. It also includes information as to the certification authority or authorities associated with a certificate management node.

(b) PKI node privilege information. When a subdomain of the PKI is hierarchically structured, the privileges of a node may be administered by a node's superior management node. The node privilege information indicates which sensitive optional functions a node may perform and any configuration restrictions.

(c) PKI operator privilege information. This includes information on individual operators, and their privileges, at each certificate management node. In some cases, operator privileges will be entirely local to a particular certificate management node. In other cases, an operator may have privileges to access information at multiple certificate management nodes.

(d) Other directory information associated with the PKI.

A distributed database management system is required to manage this information. While it might be possible to manage part or all of the above information via the X.500 directory system which is used for certificate management, it is recommended that a general-purpose distributed database management system be planned into the PKI systems.

4.8 Organizational Registration Authority Functions

An ORA is an entity whose purpose is to provide local support to a set of end entities which are physically far from their immediately superior certificate management node. An ORA performs a subset of the functions available to a certificate management node administrator responsible for directly managing a set of subordinate end entities. Such functions include:

(a) registering, de-registering, and changing attributes of subordinate end entities;

(b) carrying out identification and authentication of people associated with subordinate end entities;

(c) authorizing requests for confidentiality key recovery or certificate recovery;

(d) accepting and authorizing requests for certificate revocation;

(e) physically distributing personal tokens to, and recovering obsolete tokens from people authorized to hold them;

(f) as a possible PKI option, presenting the physical interface through which a management node initializes tokens;

(g) registering, de-registering, and assigning privileges to local ORA personnel.

An ORA has no capability to issue certificates or CRLs. An ORA is not trusted to know secret key materials of end entities. An ORA is simply a remote extension of some of the administrative capabilities of a certificate management node.

The role of an ORA in initialization of an end entity is as follows. PKI database information regarding the end entity is entered via the ORA. However, the exchange of initial key material to initialize the end entity is not, in general, relayed through the ORA system. Such exchange is accomplished via either:

(a) an on-line transaction between the end entity and certificate management node or

(b) by physical transfer, to the end entity, of a token or other physical medium/device, which has been initialized by the certificate management node, possibly using a remoted interface at the ORA. (A less secure method would be to use postal mail, as per credit card delivery.)

The ORA may play a critical role in the transferal of a physical token, medium or device in ensuring delivery of such material to the correct person. Similarly, an ORA may play a role in delivery of a password which secures an on-line initialization transaction to the correct person.

There is no need for an ORA to have any involvement in rekeying of an end entity if such rekeying is accomplished entirely via on-line transactions. Such transactions can occur directly between end entity and certificate management node.

This ORA role differs in one respect from that proposed in [NIS1], which suggests that an ORA acts as an intermediary in all communications between end entity and certification authority. The latter approach would require an ORA to be trusted with end entity key materials. This is considered both unnecessary and undesirable for the PKI, from a functional and a cost perspective. The undesirability lies in trust requirements in ORA platforms and in performance and availability penalties.

An ORA will require its own encryption and digital signature capabilities hence will incorporate the functions of both an encryption entity and a digital signature entity.

5. Miscellaneous Design Topics

5.1 Interoperation with External PKIs

There is a requirement for interoperability between the federal government PKI and the PKIs of allied countries, financial institutions, industry (U.S. and international), network service providers, and other tiers of government in the U.S. Such interoperability will permit a digital signature generated in a device under one infrastructure to be verified by a device under another infrastructure, and (subject to policy) may also permit encrypted communications to occur between devices under different infrastructures. For this to occur, common standards must be followed, it must be possible to construct certificate chains across the interoperating infrastructures, and there must be a means of obtaining certificates and CRLs from other infrastructures. To be able to construct the required certificate chains, it is necessary to have CA-certificates between the infrastructures (i.e., a CA from one infrastructure certifying the public key of a CA from another). Furthermore, there must be a means of coping with the existence of many different security policies and of deciding if a particular certificate chain can be trusted for a given application purpose.

5.1.1 Current Status of External Infrastructure Developments

Public-key infrastructure developments in the external environments of interest are as follows:

(a) Since 1993, the Canadian Communications Security Establishment (CSE) has been conducting a concerted planning activity for a public key infrastructure called the Designated/Electronic Commerce Canadian Electronic Key Management System. The technical recommendations from that work are reflected in [BNR1]. Having common roots, the directions of that activity generally align with the recommendations of this report.

(b) In the U.K. and Europe, several research or study projects have been established in this field. Examples are:

- In July 1993, the Commission of the European Communities issued a work plan on Electronic Signatures and Trusted Third Party Services, accompanied by a Call for Proposals for a feasibility study on these services.

- In October 1993, the Commission of the European Communities published a draft "Green Book" on the Security of Information Systems, as a major step in the development of an "action plan" to establish information systems security programs, including planning for Europe-wide digital signature and trusted third party services [CEC1].

- The European PASSPORT research project developed a pilot X.509-based authentication system [ROE1].

None of these activities give a good indication of likely long-term European directions.

(c) In the financial industry, the leading relevant activity is the work of the ANSI X9F1 committee on U.S. national standards for the use of public-key techniques in banking. The major standards being developed are:

- The ANSI X9.30 standard on *Public Key Cryptography Using Irreversible Algorithms for the Financial Services Industry*, which includes parts 1 and 2 reflecting the DSS and SHA specifications, respectively.

- The ANSI X9.31 standard on *Public Key Cryptography Using Reversible Algorithms for the Financial Services Industry*, which includes part 1 on RSA signatures and part 2 on hash functions other than SHA.

- The ANSI X9.57 standard on Certificate Management (this was originally part 3 of draft X9.30).

- The ANSI X9.55 standard on X.509 Certificate Extensions, which is generally aligned with the ISO/IEC DAM. This standard is targeted for eventual folding into X9.57.

The work on certificate management in this committee is new and original, and is leading the development of public-key infrastructures for the financial industry worldwide.

(d) Ongoing work in the Internet Engineering Task Force (IETF). The pre-1993 work on Internet Privacy Enhanced Mail [BAL1, KAL1, KEN1, LIN1] was significant because it led practical experimentation on public-key infrastructures in multiple-security-domain environments. (See 4.1 for further discussion of the PEM public-key infrastructure.) More recently, a new project has been established on Public-Key Infrastructure (X.509), in a new Working Group called the PKIX group. This group will profile X.509 v3 for Internet applications and

will do other necessary supporting standardization such as the development of required management protocols.

5.1.2 External PKI Standards Evolution

The three technical-standardization activities that are expected to most influence the shape of future external public-key infrastructures are:

(a) The ISO/IEC project extending X.509 to v3 certificates and v2 CRLs.

(b) The ANSI X9F1 work identified in 5.1.1 (c).

(c) The Internet Engineering Task Force (IETF) PKIX project identified in 5.1.1 (d).

It is recommended that every attempt be made to align with these ISO/IEC, ANSI X9F1, and Internet activities. The federal government should promote broad adoption of the X.509 v3 certificate format and should seek collaboration with other allied governments, with the goal of ensuring that different government and private infrastructures will ultimately interoperate with flexibility.

5.1.3 Interoperation with the MISSI Infrastructure

The future relationship between the federal PKI and the MISSI Phase 1 infrastructure is not yet clear — these infrastructures may evolve to become one and the same or they may be considered interoperating infrastructures. In any case, the MISSI Phase 1 infrastructure design is expected to evolve from its current prototype design [NSA1, NSA2] in directions which will facilitate interoperation.

This subsection aims to identify the main differences between the PKI design recommended in this report and the current MISSI Phase 1 prototype design, and to recommend possible directions to facilitate ultimate integration or interoperation.

The primary differences between the PKI and MISSI Phase 1 infrastructures are:

(a) The MISSI Phase 1 infrastructure is limited to use with the DSS, KEA, and EES algorithms. The broader PKI may need to be more general, in allowing defacto algorithms to also be used.

(b) The MISSI Phase 1 infrastructure is limited to use with the Fortezza PCMCIA token. The PKI is more general in that it will support software implementations of end entities in addition to PCMCIA tokens, including but not limited to Fortezza.

(c) The MISSI Phase 1 infrastructure uses the X.509 public-key information field in a non-standard way. In addition to conveying a public key, this field is overloaded with subsidiary information including access control information and, optionally, two distinct public keys (one for DSS and one for KEA).

(d) The MISSI Phase 1 infrastructure imposes restrictions on X.500 naming within its supported domains, including restrictions on sizes of name components and a name subordination restriction.

(e) The MISSI Phase 1 infrastructure assumes a fully hierarchical trust structure. All certificate chains start with a Root Authority (PCA) or Root Registration Authority (PAA). In the proposed PKI, certificate chains may start at any certification authority superior to the certificate-using system.

(f) The MISSI Phase 1 infrastructure introduces the concept of a compromised key list (CKL) or key-material revocation list (KRL), which is a MISSI-wide list of key-materials which have been revoked owing to suspected key compromise. CKLs constitute another means of advising of revocations beyond the standard X.509 CRLs. CKLs are compiled and distributed by Root Authorities (PCAs).

Accommodating points (a) and (b) has different ramifications in the cases of digital signature and encryption support. For digital signatures, the DSS algorithm will be available for use throughout the PKI proper, through implementation in software-based digital signature entities, Fortezza tokens, and commercial (i.e., non-Fortezza) tokens. It is highly desirable that these digital signature entities be able to interoperate with MISSI Fortezza-based digital signature entities, i.e., a DSS digital signature generated by a Fortezza-based or non-Fortezza-based entity can be verified by either a Fortezza-based or non-Fortezza-based entity. This can be achieved provided the certification infrastructures are inter-linked (discussed further below).

For encryption purposes, interoperation is restricted to holders of Fortezza tokens or other approved (hardware) implementations of EES and the SKIPJACK algorithm.

Point (c) above constitutes a restriction resulting primarily from MISSI use of the X.509 v2 certificate format. The PKI will avoid such restrictions by employing the v3 certificate format. If MISSI also migrates to use of the v3 format, compatibility between PKI and MISSI can result. Should MISSI continue to use the v2 format, interoperability can still occur with those end entities which support the MISSI format. However, this would necessitate building into certificate management nodes support for the special MISSI algorithm identifier, and associated public key information format.

Point (d) above also constitutes a restriction resulting from MISSI use of the X.509 v2 certificate format, in conjunction with the PEM name subordination rule. The naming restrictions can be avoided through the use of the v3 format and the constraint

extensions. Should MISSI not move to v3 and relax this restriction, prospects of interoperability of MISSI with the broader PKI would be at risk because of the inability to map to natural X.500 naming structures.

Point (e) above represents a fundamental MISSI restriction which needs to be relaxed to achieve broader interoperation. The X.509 v3 certificate format, with its policy and constraint extensions, represents the obvious path to resolution of this issue.

Point (f) above relates to a technical/standardization issue. The MISSI CKL approach satisfies an important requirement, namely, the requirement to expeditiously distribute revocation information regarding compromised keys to potential public key users. However, this approach depends upon the use of secure e-mail for distributing CKLs. While this is practical when supporting a secure e-mail application (e.g., the U.S. Defense Messaging System), it can present implementation problems in supporting non-e-mail applications. The requirement for partial CRLs containing only compromised key revocations is currently being addressed by the ANSI and ISO/IEC projects on certificate extensions. Rather than depend on secure revocation list distribution, this approach will use directory-based distribution. It is preferable that the PKI adopt this standard approach.

5.1.4 Certificate Formats for PKI/MISSI Interoperation

As discussed in 5.1.3, the certificate formats currently used by MISSI [NSA2] are non-standard, in that the SubjectPublicKeyInfo field is overloaded with several data items beyond the public key value. To achieve full interoperability between the two infrastructures, it is desirable to work towards agreement on use of X.509 v3 certificate extension fields for the data items which were not accommodated by the v2 format. Following is a suggested starting point, based on the latest ISO/IEC certificate extensions Proposed Draft Amendment [ISO/IEC DAM].

From the federal PKI perspective it is preferable to have distinct certificates for DSS and KEA keys. Therefore, it is recommended that interoperation with MISSI be limited to use of single-key certificates.

In adopting X.509 v3 certificates, it is recommended that the special fields in the MISSI prototype certificate format be accommodated as follows:

(a) *Version*: Not needed, as version can be implied by the algorithm component of SubjectPublicKeyInfo.

(b) *Type*: Not needed, as type can be implied by the algorithm component of SubjectPublicKeyInfo.

(c) *KMID*: Conveyed in the keyIdentifier subfield of the Key Attributes extension.

(d) *Clearance*: Conveyed in a specially-defined Directory attribute in the Subject Directory Attributes extension. The clearance may be considered sensitive data, in which case distribution within a public-key certificate is not appropriate anyway and use of ANSI attribute certificates should be considered.

(e) *Privileges*: Conveyed in a specially-defined Directory attribute in the Subject Directory Attributes extension, or in a non-critical private certificate extension, which may be ignored by non-MISSI applications.

(f) *Public Key Length*: Not needed, as each public key is in its own ASN.1 BIT STRING encoding.

(g) *Public Key*: Assuming the two-key certificate option is dropped by MISSI, one public key will be conveyed in the SubjectPublicKeyInfo field of the certificate. If the two-key certificate continues to be used by MISSI, the DSS key would be conveyed as the primary key and the KEA public key would be conveyed in a separate private certificate extension.

5.2 Communication Protocols

This section discusses the communications protocols required to support on-line communications among certificate management nodes, organizational registration authorities, end entities, and directory services.

5.2.1 Directory Access Protocols

The primary role of directory services in the PKI is the distribution of public-key certificates and certification revocation lists to end entities or other systems requiring them. Directory services used for this purpose do not require a high level of trust because certificates and CRLs do not require confidentiality and provide their own integrity internally.

For directory services, the PKI will employ the X.500 architecture and X.500-based protocols. Use of these protocols will span directory services operated by federal departments and agencies, by federal infrastructure service providers, and by the operating authorities of PKI nodes. In general, requests for certificates or CRLs will be chained to the appropriate DSA in any of these directory services. Provision should also be made for replicating non-sensitive directory entry information in DSAs of different directory services. In particular, entry information for all PKI certification authorities should be replicated to any directory service requesting it. If such replication is in place, it will not be necessary to support chaining of requests from low assurance directory services to the more sensitive DSAs of PKI operating authorities.

For end entity access to directory services, both of the following protocols should be supported:

- The X.500 Directory Access Protocol (DAP) [ISO/IEC 9594] which operates over a full OSI protocol stack; and
- The Internet Lightweight Directory Access Protocol (LDAP) [YEO1] which operates over a TCP/IP protocol base.

These protocols are functionally equivalent,. However, it can be anticipated that some end-entity products will support DAP and others will support LDAP.

The preferred protocol for use in directory replication is the X.500 Directory Information Shadowing Protocol (DISP) [ISO/IEC 9594].

5.2.2 End Entity Management Protocols

On-line transactions are potentially required for a certificate management node to interact with end entities for the following purposes:

(a) *End-entity initialization*: On-line transactions are typically required to initialize software-based end entities. For a token-based end entity, on-line initialization is generally unnecessary because initial loading and delivery of the physical token achieves the required initialization.

(b) *Public-private key pair update*: For both software-based and token-based end entities, on-line key pair update exchanges are typically required.

(c) *Recovery of end-entity key materials*: Following loss of end-entity information, e.g., as a result of a forgotten password or PIN, an on-line transaction may be required as part of the recovery process.

(d) *Audit trail data uploading*: Provision may be made to upload audit data from end entities.

There are no well-established standards for the above protocols. Hence, the PKI will potentially need to support different protocol exchanges for different encryption or digital signature products. The federal government should work towards establishment of a public standard for end-entity management protocol exchanges.

Specifications which might provide a suitable basis for such a standard are:

- The ISO/IEC Generic Upper Layers Security (GULS) standard [ISO/IEC 11586] defines a generic security exchange protocol which can be used to support such protocol exchanges. Using GULS, a complete specification for protected key-material exchanges can be produced relatively easily and either formally standardized or privately registered by the government.

- Northern Telecom has specified, for its Entrust product, a protocol called the Secure Exchange Protocol (SEP). This is a GULS-based protocol which includes definitions of security exchanges to initialize a software end-entity, update key pair(s) for an end entity, and perform key-material recovery for an end entity.

- The management protocol being developed as part of the MISSI program.

Protocols for audit trail data uploading, as identified in (d) above, are a low priority requirement, because commercial end entity products do not generally support such a capability. As a longer term activity, the federal government should consider promoting establishment of protocol standards for audit trail uploading. The following specifications might provide a useful basis for such a standard:

- An extended version of the management protocol discussed above.

- Internet SNMP [CAS1].

- The OSI CMIP protocol [ISO/IEC 9595, ISO/IEC 9596] for which object definitions for conveying audit trail records already exist [ISO/IEC 10164-8].

5.2.3 Organizational Registration Authority Protocols

Communications between an ORA and its management node constitute a protected session of administration transactions. The set of transactions supported should include transactions for:

(a) registering, de-registering, and changing attributes of subordinate end entities;

(b) requesting key material recovery for a subordinate end entity;

(c) requesting certificate revocation for a subordinate end entity;

(d) querying and updating databases which reflect physical token management;

(d) registering, de-registering, and assigning privileges to local ORA personnel.

The transactions between an ORA and its management node should be protected for both confidentiality and integrity using an appropriate session-oriented protection protocol, such as protected remote procedure calls.[11]

5.2.4 Management Node to Management Node Communications

On-line communications between a certificate management node and its subordinate certificate management nodes need to support various types of information transfer, including:

(a) transfer of directory entry information, including public-key certificates and certificate revocation lists;

(b) sensitive key-material exchanges, including protocol exchanges for such purposes as node key-pair establishment and update;

(c) distributed database management;

(d) audit trail data uploading.

For (a), standard directory access protocols as discussed in 5.2.1 should be employed. However, it should be recognized that certain directory access operations between certificate management nodes may be sensitive. Because X.500 protocols have no in-built confidentiality protective features, it may therefore be necessary to operate these protocols over secured connections.

For (b), an adaptation of the protocol for end-entity management discussed in 5.2.2 would serve best.

For (c), there is a need for a protocol to support sensitive distributed database accesses between certificate management nodes. It is assumed that the PKI will use a commercial database management system. Such system may provide the necessary protective features. If not, the requisite protection will need to be provided by operating the database protocol over secure sessions.

For (d), an appropriate protocol will need to be determined. Standardization is not an important requirement because this protocol is used only between certificate management nodes. This requirement can possibly be satisfied by the distributed database management facilities of (c).

[11] A suitable mechanism for such protection is the Internet Simple Public Key Mechanism (SPKM) for use with the Internet Generic Security Service API (GSS-API). The SPKM specification is currently being progressed to a proposed Internet standard.

5.3 Node Operator Interface

Each certificate management node will require an operator interface giving access to such operator functions as:

(a) registering, de-registering, and changing attributes of subordinate encryption entities and digital signature entities;

(b) registering, de-registering, and changing attributes of subordinate certificate management nodes (including maintenance of node privilege information, if applicable);

(c) registering, de-registering, and assigning privileges to local certificate management node personnel;

(d) loading physical media/devices, or generating initial key material, to support end-entity initialization;

(e) initializing tokens;

(f) authorizing requests for confidentiality key recovery or certificate recovery;

(g) invoking confidentiality key recovery or certificate recovery;

(h) initiating certificate revocation;

(i) maintaining an operator's password;

(j) managing the key backup/archival system;

(k) processing and archiving audit trails;

(l) responding to security alarms.

At the central archive facility, there will be certain additional functions associated with managing a central certificate/CRL archive system.

For each type of command, a specific privilege level will be required. Every individual operations person will be assigned a set of privileges. Assignment of privileges will be in accordance with certain PKI policy stipulations, plus local policy rules. Administration of privileges can be entirely local to the node concerned or, dependent on policy, can be wholly or partially controlled by a superior node.

For certain of the more sensitive commands, there may be a requirement that two or more personnel with appropriate privileges must be simultaneously present to issue the command. Node software should be able to enforce this requirement.

Dependent upon operational environment, appropriate authentication mechanisms (e.g., password-based, token-based) for operations personnel will be required.

References

[ANSI X9.30] ANSI, *American National Standard, Public-Key Cryptography Using Irreversible Algorithms for the Financial Services Industry* .

[ANSI X9.31] ANSI, *American National Standard, Public-Key Cryptography Using Reversible Algorithms for the Financial Services Industry* .

[BAL1] D. Balenson, *Privacy Enhancement for Internet Electronic Mail, Part III: Algorithms, Modes, and Identifiers*, Internet RFC 1423, 1993.

[BNR1] Bell-Northern Research, *Designated/Electronic Commerce Canadian Electronic Key Management System Project Definition: Final Report* (Prepared for Canadian government under Contract W2213-4-4657/04-QE), February 28, 1995 (Restricted distribution).

[CAS1] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, *Introduction to Version 2 of the Internet-standard Network Management Framework*, Request for Comments (RFC) 1441, Internet Activities Board, 1993.

[CEC1] Commission of the European Communities, *Green Book on the Security of Information Systems*, Draft 4.0, October 1993.

[CON1] U.S. Congress, Public Law 100-235 — Jan. 8, 1988, Computer Security Act of 1987.

[DIF1] W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, vol. IT-22, no. 6 (1976), pp. 644-654.

[FIPS1] U.S. Department of Commerce, *Digital Signature Standard*, Federal Information Processing Standards Publication FIPS PUB 186, 1994.

[FIPS2] U.S. Department of Commerce, *Secure Hash Algorithm*, Federal Information Processing Standards Publication FIPS PUB 180, 1993.

[FIPS3] U.S. Department of Commerce, *Escrowed Encryption Standard*, Federal Information Processing Standards Publication FIPS PUB 185, 1994.

[FIPS4] U.S. Department of Commerce, *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards Publication FIPS PUB 140-1, 1994.

[FOR1] W. Ford, *Computer Communications Security: Principles, Standard Protocols and Techniques*, Prentice-Hall, 1994.

[FOR2] W. Ford and M.J. Wiener, *A Key Distribution Method for Object-Based Protection*, Proc. 2nd ACM Conference on Computer and Communications Security, Nov. 1994.

[ISO/IEC 8824] ISO/IEC, *Information Technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)*, ISO/IEC 8824, 1993, also published as ITU-T X.680 series of Recommendations.

[ISO/IEC 9594] ISO/IEC, *Information Technology — Open Systems Interconnection — The Directory*, ISO/IEC 9594, 1993 (multi-part standard), also published as ITU-T X.500 series of Recommendations.

[ISO/IEC 9594-8] ISO/IEC, *Information Technology — Open Systems Interconnection — The Directory: Authentication Framework*, ISO/IEC 9594-8, 1993, also published as ITU-T X.509 Recommendation.

[ISO/IEC 9595] ISO/IEC, *Information Technology — Open Systems Interconnection — Common Management Information Service Definition*, ISO/IEC 9595, also published as ITU-T X.710 Recommendation.

[ISO/IEC 9596] ISO/IEC, *Information Technology — Open Systems Interconnection — Common Management Information Protocol Specification*, ISO/IEC 9596, also published as ITU-T X.711 and X.712 Recommendations.

[ISO/IEC 10164-8] ISO/IEC, *Information Technology — Open Systems Interconnection — Systems Management: Security Audit Trail Function*, ISO/IEC 10164-8, also published as ITU-T X.740 Recommendation.

[ISO/IEC 11586] ISO/IEC, *Information Technology — Open Systems Interconnection — Generic Upper Layers Security*, ISO/IEC 11586, 1995 (multi-part standard), also published as ITU-T X.830 series of Recommendations.

[ISO/IEC DTC] ISO/IEC, *Draft Technical Corrigenda to Rec. X.500 | ISO/IEC 9594 resulting from Defect Reports 9594/128 (Final text)*, ISO/IEC JTC1/SC21, August 1995.

[ISO/IEC DAM] ISO/IEC, *Draft Amendment ISO/IEC 9594 DAM-1 for Certificate Extensions*, ISO/IEC JTC1/SC21, August 1995.

[KAL1] B. Kaliski, *Privacy Enhancement for Internet Electronic Mail, Part IV: Key Certification and Related Services*, Internet RFC 1424, 1993.

[KEN1] S. Kent, *Privacy Enhancement for Internet Electronic Mail, Part II: Certificate-Based Key Management*, Internet RFC 1422, 1993.

[LIN1] J. Linn, *Privacy Enhancement for Internet Electronic Mail, Part I: Message Encryption and Authentication Procedures*, Internet RFC 1421, 1993.

[NIS1] U.S. National Institute of Standards and Technology, *Public Key Infrastructure Study*, Final Report, April 1994.

[NSA1] U.S. National Security Agency, *MISSI Phase 1 Program Overview*, Version 3.3, Oct. 17, 1994.

[NSA2] U.S. National Security Agency, *MOSAIC Key Management Concept*, Rev. 2.5, Feb. 28, 1994.

[NSA3] U.S. National Security Agency, *SDNS Message Security Protocol (MSP)*, SDN.701, Rev. 3.0, Mar. 21, 1994.

[RIV1] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, vol. 21, no. 2 (February 1978), pp. 120-126.

[ROE1] M. Roe, *PASSWORD R2.5: Certification Authority Requirements*, Cambridge University Computer Laboratory, Computer Security Group, Version 1.1, July 1993.

[YEO1] W. Yeong, T. Howes, S. Kille, *X.500 Lightweight Directory Access Protocol*, Internet RFC 1487, 1993.

Appendix A — Directory System Requirements

This appendix outlines the requirements of a U.S. federal government X.500 Directory System Agent (DSA) which is to support distribution of PKI certificates and CRLs. The relevant object class values and attributes need to be added to directory entries in the X.500 environment to enable the distribution and management of public keys. The ability to add auxiliary values to the object class attribute, as defined in the 1993 X.500 Series of Recommendations, is also required.

A.1 Directory Schema Requirements

A.1.1 Object Classes

The following object classes, as defined in ITU-T Rec. X.521 (1993 E) amended in accordance with DAM 1, shall be supported:

(a) strongAuthenticationUser;

(b) certificationAuthority-V2;

(c) cRLDistributionPoint.

Both (a) and (b) object classes are auxiliary object classes and, as such, are not intended to be instantiated as directory entries on their own, but (c) is a structural object class.

The directory entries which are to include the strongAuthenticationUser auxiliary object class will typically be entries of the structural object class organizationalPerson or other similar structural classes.

Directory entries which are to include the certificationAuthority-V2 auxiliary object class will typically be entries of the structural object class organization or organizationalUnit, or some other structural object class representing an authority. Future growth to accommodate additional object classes must also be provided.

A.1.2 Attribute Types

As specified in ITU-T Rec. X.521 (1993 E), there are mandatory attribute types which must be contained in entries which include the auxiliary object classes indicated above. These attribute types shall be supported, as defined in ITU-T Rec. X.509 (1993 E):

— userCertificate (version 3);

— caCertificate;

— certificateRevocationList;

— authorityRevocationList;

— crossCertificatePair.

Note that these attribute types may have one or more values to support confidentiality and digital signature services using one or more algorithms.

In addition, as for any use of X.500, every DN must be unambiguous/unique. In order to disambiguate at leaf nodes for entities such as organizationalPerson entries, it is strongly recommended to use a unique numeric string as an additional Relative Distinguished Name (RDN) component. Otherwise, when an entry is deleted from the directory, the DN cannot be re-used for a specific period of time. To satisfy this requirement, the serialNumber attribute (as specified in ITU-T Rec X.520) may be used. This can be added to directory entries of any object class if the DIT content rule mechanism is implemented. If this is not implemented, an additional auxiliary object class uniquelyIdentifiedEntry would enable the addition of this attribute to entries of appropriate structural object classes. Note that a given directory entry may have more than one userCertificate

A.2 Access Control Permissions

The administrator for the public key management data may be different from the administrator for other directory entry information.

The access control scheme shall be such that:

— Only the PKI administrator may add the strongAuthenticationUser or certificationAuthority values to the object class attribute for appropriate entries;

— Only the PKI administrator may modify values of the userCertificate, cACertificate, certificateRevocationList, authorityRevocationList, and crossCertificationPair attribute types;

— The PKI administrator may delete specific values of the userCertificate, cACertificate, certificateRevocationList, authorityRevocationList, and crossCertificationPair attribute types as well as delete the attribute types completely;

— The PKI administrator may delete the strongAuthenticationUser and certificationAuthority values from the object class attribute;

— The general administrator for entries which include the strongAuthenticationUser or certificationAuthority object classes may delete the entire entry for which he/she has responsibility, including the public key management object classes and attributes.

A.3 Interface Requirements

Access by both the PKI administrator for management of the public key data and by client user agents to query the DSA to obtain certificates will by via either the Directory Access Protocol (DAP) or the Lightweight Directory Access Protocol (LDAP). Both protocols should be supported.

Authentication shall initially be of type "simple". Migration to "strong" level of authentication for the PKI administrator and possibly for client users is expected in the future. UserCertificates will be signed by a PKI certification authority and verified by the directory service before being actioned.

If the on-line revocation option is to be implemented using the trusted-directory approach, a DSA supporting this revocation option will be required to support strong authentication and signed operation responses using public-private key pairs managed by the PKI.

A.4 Procedural Requirements

As the public key management data is generated in a remote manager system and deposited in the DSA, some procedures need to be established to ensure the certificates are generated for the appropriate users and that the management of certification

authority data is maintained in a timely and consistent manner.

In order to maintain synchronization between the PKI and the DSA, the following information needs to be gathered from the Government X.500 Directory Service and provided to the PKI administrator on a regular basis and in a format agreed between the relevant administrators:

— The Distinguished Names (DNs) of user entries for which a userCertificate needs to be added;

— The DNs of any strongAuthenticationUser or certificateAuthority entries which have been deleted from the directory;

— The DNs of any certificate authority entries which have been impacted by organizational changes and as a result require modifications.

The unique identifiers will be managed by the directory administrators. Each identifier is associated with one individual and never re-used. This ensures that public key data is consistent with the directory entry population and the user community represented by those entries.

A.5 Matching Rules

The certificate and CRL matching rules defined in 12.7 of [ISO/IEC DAM] are likely to be required in future procurements.

Appendix B — Certificate Revocation

B.1 General

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 distributed via a public-key infrastructure. A public-key certificate is a data structure which binds a public key bit-string to data which identifies a subject. The binding is achieved by having a trusted certification authority (CA) digitally sign the certificate. A certificate has a limited valid lifetime, indicated by a start date/time and an expiration date/time. This validity period is indicated in the signed part of the certificate. Because a certificate's signature and timeliness can be independently checked by the certificate-using client system, certificates can be distributed via untrusted communications and server systems, and can be cached in unsecured storage in certificate-using systems. While the validity period of certificates is a configurable parameter of a CA, lifetime values typically used range from several months to several years.

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

B.2 Periodic Revocation Lists

Recommendation ITU-T X.509, which is the recognized standard for public key certificate formats, defines one method of certificate revocation. This method involves each CA periodically issuing a 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, e.g., by the CA posting the CRL in its own X.500 directory entry. Each revoked certificate is identified in a CRL by its *certificate serial number*, a unique

number for the certificate which is generated by the issuing CA and included in a certificate field.

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 we usually mean the most recently-issued CRL.

A CA issues a new CRL on a regular periodic basis, e.g., hourly, daily, or weekly (the period is a configurable parameter of a CA), regardless of whether or not any new revocations have been added to the list in the latest period. This is necessary so that a certificate-using system can be certain that it has an up-to-date CRL.

An important characteristic of this revocation method is that CRLs may be distributed by exactly the same means as public-key certificates themselves, namely, via untrusted communications and server systems. This makes this approach very economical to install and to operate. (Other revocation methods described later require secure communications and trusted servers which add greatly to both system costs and operational costs.)

One limitation of this revocation method 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 certification-using systems until the next periodic CRL is issued — this may be up to (for example) one hour, one day, or one week from now. Note that there is nothing preventing a CA from generating and posting a new CRL immediately a new revocation becomes known. However, it cannot be guaranteed that such *off-cycle* CRLs will always reach certificate-using systems. For example, if an intruder deletes an off-cycle CRL from an untrusted server and leaves the previous periodic CRL in its place, this cannot be detected by the certificate-using system.

B.3 Size of Certification Revocation Lists

The size a CRL can reach is very important. With the periodic revocation list method, when a certificate-using system uses a certificate, that system generally needs to fetch a CRL and process it (including verifying the signature over the complete CRL). If CRLs can become very large, this presents performance problems in terms of both communications overheads and processing overheads in certificate-using end entities.

Entries are added to CRLs as revocations occur and an entry may be removed when the certificate expiration date is reached. The rate at which revocations occur tends to be quite unpredictable, but is clearly dependent on the size of the population of subjects

covered. The two main factors which can be used to control CRL sizes are therefore the size of the subject population and the certificate validity period.

It is highly undesirable to force certificate lifetimes to be short in order to limit CRL sizes. Short certificate lifetimes have a number of negative ramifications, such as higher operational costs, user inconvenience, and higher demands on archive resources.

In the 1988 and 1993 versions of X.509, each CA had two CRLs — one for all the end-user subjects it certified and one for the other CAs which it certified. The latter list could be expected to be very short — usually empty. This was very valuable in reducing concerns of CRL processing overheads when verifying certificate chains — of all the certificates in the chain, only the one end-user certificate would be likely to encounter associated CRL size problems. However, the concern remained that the CRL for the end-user certificate needed to cover the entire population of end-users for one CA. It is highly 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 an entirely unacceptable size.

With the X.509 certificate and CRL extensions being introduced in the 1994-1995 time frame, this problem is being resolved. With the new certificate and CRL formats, it is 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* which has its own CRL and its own directory entry for distributing that CRL. Therefore, the maximum size to which a CRL can grow can be controlled by a CA. These X.509 extensions also permit separate CRL distribution points to 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 may be updated more frequently than the former.

B.4 Broadcast Revocation Lists

The periodic revocation list approach described above is sometimes called a *pull* method of CRL distribution, because a certificate-using system fetches CRLs from a directory when it needs them. An alternative distribution approach is a *push* method, in which a CA broadcasts revocation lists to certification-using systems as new revocations are posted. Such broadcasts are accomplished via protected communication means such as secure e-mail or a protected transaction protocol. The major advantage of this approach is that "important" revocations, e.g., revocations resulting from key compromise or revocations of CA-certificates, can be distributed very quickly, without the time granularity delay problem inherent in the periodic revocation list approach.

However, there are several potential problems with this approach. First is the requirement for a protected distribution method to ensure that CRLs reach their intended destinations. Second is the massive amount of traffic which would be generated if this

method were to be used to notify of all revocations — it is practical only for notifying of more critical revocations. Third is the problem of deciding, in an arbitrarily large infrastructure environment, which revocations need to be advised to which certificate-using systems. Fourth is the absence of standards for such methods, which mitigates against their widespread use.

The DoD MISSI Phase 1 program, which provides a public-key infrastructure for the Defense Messaging System (DMS), presents a good example of use of this method. MISSI uses a conventional periodic revocation list approach to advise of revocations generally. However, it has an additional broadcast-style method for advising of the relatively small number of revocations resulting from compromised keys. The latter method uses an additional revocation list called a compromised key list (CKL) or Key Revocation List (KRL). In essence, one central CKL is maintained for an entire root domain network, on the basis of compromise notifications advised by the various CAs in the network. The CKL is distributed via secure e-mail.

In the DMS case, all of the potential problems with broadcast revocation lists are averted. Availability of a protected distribution method is not a concern because *the* target application being supported is secure e-mail, hence all systems involved have ready access to secure e-mail functionality. Because the broadcast approach is used only for notifying key compromises and because such compromises can be expected to be very rare in the well-controlled, Fortezza-based DMS environment, the CKL is unlikely to ever gain significant size. Because one CKL is used for the entire network, there is no issue as to who should receive which CKL.

However, the MISSI CKL approach does not scale well nor generalize into other application environments. Therefore, it does not give us a ready-made revocation approach suitable for commercial application in the international, or even national, arena.

B.5 Immediate Revocation

One concern often expressed with the periodic revocation list approach is that certificate-using systems cannot tolerate the delay in revocation notification resulting from the time granularity factor. A great deal of damage can be caused from a compromised key in a period like a day. In an ideal world, knowledge of a revoked certificate would be made available to a certificate user immediately that user wants to use that certificate. One may therefore consider ways in which *immediate revocation* might be achieved in public-key infrastructures.

Direct immediate revocation would require a certificate-using system to execute an on-line transaction with the CA issuing a certificate to confirm its validity. The transaction must be secured, to the extent that its timeliness and its source must be assured to the certificate-using system. This requires, as a minimum for each transaction, generation a of a digital signature by the CA and verification of that signature by the certificate-using

system. The CA must operate a high-availability on-line service, accessible by all potential users, and this service must be provided from within a secured environment.

While this may be quite achievable in a restricted environment, e.g., a closed community of certificate-subjects and certificate-users based on a single CA, it presents several problems. One problem is the requirement for the trusted on-line server — its acquisition and operational costs. In particular, recognizing that this server needs to generate one digital signature per query transaction, the processing overheads are likely to demand very costly cryptographic processor resources. Another problem is that application of this approach to a large-scale infrastructure would only be possible if new standards are established to support the approach and if all CAs in the infrastructure commit to implementing and supporting it. Adoption of a direct immediate-revocation approach for large-scale infrastructures therefore seems very unlikely in the foreseeable future.

Are there any other revocation approaches which would move us closer to the immediate-revocation goal while averting the problems inherent in the direct approach? Following are some possibilities.

(a) *Directory updating*: A simple approach is for the CA to immediately remove the certificate from the subject's directory entry when a certificate is revoked; those certificate-using systems which are particularly concerned with timeliness perform a fresh certificate retrieval whenever they wish to use a certificate. While this is comparatively simple to implement, it is not, in general, secure because directory servers and directory communications are not generally considered trusted. For example, an intruder in the communications path could insert an old copy of a certificate into a directory retrieval transaction, even if that certificate had subsequently been revoked.

(b) *Trusted directory*: To add strength to the directory updating approach, we could ensure that the directory service agent (DSA) which supplies the certificate is implemented as a trusted system and that the transaction response returning the certificate is digitally signed by that DSA and contains a current time-stamp. The X.500 directory protocols contain an optional feature called *signed operations* which could support the required protection of the transaction response. This is a reasonable way of achieving immediate revocation, provided all X.500 components involved implement the signed operations option and provided the certificate-using system has confidence that the DSA is indeed operated under trusted conditions and under the policy of the CA involved.

(c) *Fine-granularity periodic CRLs*: Given that the issue cycle of periodic CRLs is a CA decision, is it possible to make this period sufficiently short that revocation notifications are timely enough using the basic periodic CRL approach without requiring trusted servers or protecting protocols? There is no fundamental problem in setting the CRL issue cycle to 1 day, 1 hour, or 10 minutes, provided the lists are not so large that processing and communications overheads become

unacceptable and provided the directory system is able to cope with the distribution. Such granularity delays are likely to be perfectly adequate for many applications — with such applications, there may typically be a delay of hours or even days before a CA is notified of a suspected compromise, hence the CRL granularity delay may be insignificant. While fine-granularity CRL cycles are probably infeasible with the original X.509 formats because of CRL size problems, the 1995 X.509 revision allows separate CRLs to be used for different subpopulations and for different revocation reasons, e.g., routine revocations and key compromises. It is therefore possible to engineer an infrastructure such that the CRLs covering compromise situations can be kept small and can be issued with a very fine-granularity cycle, e.g., hourly or even more frequently. CRLs for routine revocations, which are likely to be much larger, may be issued with a coarser granularity, e.g., daily.

If immediate revocation is considered a requirement, the fine-granularity CRL approach and the trusted directory approach are the two approaches most worthy of consideration.