- **April 21, 2005 -- Draft Special Publication 800-57, Recommendation for Key Management**

  Part 1 (General):

  Part 2 (Best Practices for Key Management Organizations):

  Drafts of NIST Special Publication 800-57 *Recommendation for Key Management, Parts 1 and 2* are available for public comment. This Recommendation provides cryptographic key management guidance. Part 1 provides guidance and best practices for the management of cryptographic keying material. Part 2 provides guidance on policy and security planning requirements for U.S. government agencies.

  Comments will be accepted on Part 1 until June 3, 2005. Please send comments to Key_mgmt @ nist.gov, with "Comments on SP 800-57, Part 1" in the subject line.

  Comments will be accepted on Part 2 until May 18, 2005. Please send comments to Key_mgmt @ nist.gov, with "Comments on SP 800-57, Part 2" in the subject line.

SPECIAL PUBLICATION 800-57

RECOMMENDATION FOR KEY MANAGEMENT

Part 1: General Guideline

Since this document is under development, the reader should be aware that portions of the document may change during the continuing development process.

NIST requests that comments on this document be provided by April 3, 2003, although comments will be accepted at any time. Please send comments to GuidelineComments@nist.gov. Since two parts of this draft Recommendation are being posted for comment, please indicate which document is being commented on. Thanks.

This document contains the following specific questions that NIST would like feedback on:

Page 96, Section 8.8.1, no. 3: *Are there any other applications for which collisions are a concern*?

Page 96, Section 8.8.1, no. 4: *Are there other applications that do not have collision concerns*?

Page 99, Section 8.8.2: *For 112 bit of security, should the output of SHA-256 be truncated to 224 bits, should truncation not be allowed, or should truncation be optional*?

Page 100, Section 8.8.3, no 4: *Are there other combinations to be considered*?

Page 122, Appendix C:*Appendix C has been provided to invite comment on the use of passwords for the generation of cryptographic keys. Some or all of this material may be provided in another document at a later time.*

Table of Contents

KEY MANAGEMENT GUIDELINE

Part 1: General Guidance

1. INTRODUCTION

Cryptographic mechanisms are one of the strongest ways to provide security services for electronic applications and protocols and for data storage. The National Institute of Standards and Technology (NIST) publishes Federal Information Processing Standards (FIPS) that specify cryptographic techniques for protecting sensitive unclassified information.

Since NIST published the Data Encryption Standard (DES) in 1977, a suite of approved standardized algorithms has been growing. New classes of algorithms have been added, such as secure hash algorithms and asymmetric key algorithms for digital signatures. The suite of algorithms now provides different levels of cryptographic strength through a variety of key sizes. The algorithms may be combined in many ways to support increasingly complex protocols and applications. While a FIPS is mandatory for U.S. government agencies using cryptography for the protection of their sensitive unclassified information, a FIPS may also be followed, on a voluntary basis, by other organizations that want to implement sound security principles in their computer systems.

The proper management of cryptographic keys is essential to the effective use of cryptography for security. Keys are analogous to the combination of a safe. If the combination becomes known to an adversary, the strongest safe provides no security against penetration. Similarly, poor key management may easily compromise strong algorithms. Ultimately, the security of information protected by cryptography directly depends on the strength of the keys, the effectiveness of mechanisms and protocols associated with keys, and the protection afforded the keys. Cryptography can be rendered ineffective by the use of weak products, inappropriate algorithm pairing, poor physical security, and the use of weak protocols.

All keys need to be protected against modification, and secret and private keys need to be protected against unauthorized disclosure. Key management provides the foundation for the secure generation, storage, distribution, and destruction of keys. Another role of key management is key maintenance, specifically, the update/replacement of keys.

1.1 Goal/Purpose

Users and developers are presented with many new choices in their use of cryptographic mechanisms. Inappropriate choices may result in an illusion of security, but little or no real security for the protocol or application. Basic key management guidance is provided in [SP800-21]. This guideline (i.e., SP 800-57) expands on that guidance, provides background information and establishes frameworks to support appropriate decisions when selecting and using cryptographic mechanisms.

1.2 Audience

The audiences for the *Key Management Guideline* include system or application owners and managers, cryptographic module developers, protocol developers, and system administrators. The guideline has been provided in three parts. The different parts into which the guideline has been divided have been tailored to specific audiences.

Part 1 of the document provides general key management guidance that is intended to be useful to both system developers and system administrators. Cryptographic module developers may benefit from this general guidance through a greater understanding of key management features required to support specific intended ranges of applications. Protocol developers may identify key management characteristics associated with specific suites of algorithms and gain a greater understanding of the security services provided by those algorithms. System administrators may use this document to determine which configuration settings are most appropriate for their information.

Part 2 of the guideline is tailored for system or application owners for use in identifying appropriate organizational key management infrastructures, establishing organizational key management policies, and specifying organizational key management practices and plans.

Part 3 of the guideline is intended to provide guidance to system administrators regarding the use of cryptographic algorithms in specific applications, select products to satisfy specific operational environments, and configure the products appropriately.

Though some background information and rationale are provided for context and to support the guideline's recommendations, the guideline assumes that the reader has a basic understanding of cryptography. For background material, readers may look to a variety of NIST and commercial publications. [SP800-21] includes a brief introduction to cryptography. [SP 800-5] and [IPKI] provide an introduction to public key infrastructure. A mathematical review of cryptography and cryptographic algorithms is found in [HAC] and [AC].

1.3 Scope

This guideline encompasses cryptographic algorithms, infrastructures, protocols, and applications, and the management thereof. All cryptographic algorithms currently approved by NIST for the protection of unclassified but sensitive information are in scope. Cryptographic infrastructures that are commonly used to distribute keys for approved algorithms are considered, such as Kerberos and the X.509 public key infrastructure. The guideline addresses those protocols and applications that are widely used by Federal agencies (e.g., such as TLS).

This guideline focuses on issues involving the management of cryptographic keys: their generation, use, and eventual destruction. Related topics, such as algorithm selection and appropriate key size, cryptographic policy, and cryptographic module selection, are also included in this guideline. Some of the topics noted above are addressed in other NIST standards and guidance. For example, a public key infrastructure is addressed in [IPKI]. This guideline supplements more focused standards and guidelines.

This guideline does not address implementation details for cryptographic modules that may be used to achieve the security requirements identified. These details are addressed in [SP 800-21], [FIPS 140-2], and the derived test requirements (available at http://csrc.nist.gov/cryptval/).

This guideline also does not address the requirements or procedures for operating an archive other than discussing the types of keying material that are appropriate to include in an archive and the protection to be provided to the archived keying material.

This guideline often uses “requirement” terms; these terms have the following meaning in this document:

1. **shall**: This term is used to indicate a requirement of a Federal Information Processing Standard (FIPS) or a requirement that must be fulfilled to claim conformance to this Recommendation. Note that **shall** may be coupled with **not** to become **shall not**.
2. **should**: This term is used to indicate very important guidance. While the guidance may not be stated in a FIPS, ignoring the requirement could result in undesirable results. Note that **should** may be coupled with **not** to become **should not**.

1.4 Relationship to FIPS

FIPS security standards are valuable because:

1. They establish an acceptable minimal level of security for U.S. government systems. Systems that implement these standards offer a consistent level of security approved for sensitive, unclassified government data.
2. They often establish some level of interoperability between different systems that implement the standard. For example, two products that both implement the Advanced Encryption Standard (AES) cryptographic algorithm have the potential to interoperate, provided that the other functions of the product are compatible.
3. FIPS standards often provide for scalability because the U.S. government requires products and techniques that can be effectively applied in large numbers.
4. FIPS are scrutinized by the U.S. government to assure that they provide an adequate level of security. This review is performed by U.S. government experts in addition to the reviews performed by the public.
5. FIPS techniques are re-assessed every five years for their continued effectiveness. If any technique is found to be inadequate for the continued protection of government information, the standard is revised or discontinued.
6. Several of the cryptography-based FIPS (e.g., DES, TDES, SHA-1, DSA, and Cryptographic Modules) have required conformance tests. These tests are performed by accredited laboratories on vendor products that claim conformance to the standards. Vendors are permitted to modify non-conforming products so that they meet all requirements. Users of validated products can have a high degree of confidence that validated products conform to the standard.

Since 1977, NIST has built up a standards "toolbox" of FIPS security standards that form a basis for the implementation of strong cryptography. This guideline refers to many of those standards and provides guidance on how they may be properly used to protect sensitive information.

1.5 Content and Organization

The guideline is written for several different audiences and is divided into three parts.

Part 1, *General Guidance*, contains basic key management guidance. It is intended to advise developers and system administrators on the "best practices" associated with key management.

1. Section 1, *Introduction*, establishes the purpose, scope and intended audience of the Key Management Guideline
2. Section 2, *Glossary of Terms and Acronym*, provides definitions of terms and acronyms used in this part of the Key Management Guideline. The reader should be aware that the terms used in this guideline may be defined differently in other documents.
3. Section 3, *Security Services*, defines the security services that may be provided using cryptographic mechanisms.
4. Section 4, *Cryptographic Algorithms*, provides background information regarding the cryptographic algorithms that use cryptographic keying material.
5. Section 5, *Protection Requirements for Cryptographic Information*, classifies the different types of keys and other cryptographic information according to their functions, specifies the protection that each type of information requires and identifies methods for providing this protection. These protection requirements are of particular interest to cryptographic module vendors and application implementers.
6. Section 6, *Key States*, identifies the states in which a cryptographic key may exist during its lifetime.
7. Section 7, *Key Management Functions*, identifies the multitude of functions involved in key management. This section is of particular interest to cryptographic module vendors and developers of cryptographic infrastructure services.
8. Section 8, *General Key Management Guidance*, discusses a variety of key management issues related to the keying material identified in Section 5. Topics discussed include key usage, cryptoperiod length, domain parameter validation and public key validation, accountability, audit, key management system survivability, guidance for cryptographic algorithm and key size selection, and specific guidance for the use of the key establishment schemes identified in [SP 800-56].

Part 2, *General Organization and Management Requirements*, is intended primarily to address the needs of system owners and managers. It provides a framework and general guidance to support establishing cryptographic key management within an organization and a basis for satisfying key management aspects of statutory and policy security planning requirements for Federal government organizations. Section 1, *Introduction*, discusses the organization of Part 2. Section 2, *Key Management Infrastructures*, describes a generic key management infrastructure. The infrastructure described is an adaptation of the Public Key Infrastructure (PKI) and other widely employed key management infrastructures to provide a Key Management Infrastructure for the management of both public and secret keying material in support of a broad range of cryptographic services. Section 3, *Key Management Policy and Practices*, provides guidance for the development of organizational key management policy statements, key management practices statements, and security plans that may be needed in support of institutional use of cryptography. Section 4, *Information Technology System Security Plans*, suggests key management planning and requirements that may be appropriate for inclusion in Major Applications Security Plans and General Support Systems Security Plans for major systems that apply cryptography. Section 5, *Key*

Management Plans for Cryptographic Devices or Applications, identifies planning and documentation that are useful for describing the set of key management products and services that may be required by a cryptographic device or application throughout its lifetime.

Part 3, *Implementation-Specific Key Management Guidance*, is intended to address the key management issues associated with currently available implementations.

3.1 Section 1, *Selected Infrastructures*, is provides guidance on key management issues associated with widely deployed key establishment infrastructures. This section addresses key management requirements for both the infrastructure and its users. This section is of particular interest to parties developing or deploying infrastructures for key establishment. System and application owners may use this section to determine whether a particular infrastructure supports the security services required for their information.

3.2 Section 2, *Selected Protocols*, provides guidance in the use of common cryptographic protocols. Developers may use this section to identify important implementation details. System and application owners may use this section to establish configuration parameters appropriate for their information.

3.3 Section 3, *Selected Applications*, provides guidance on the use of cryptographic algorithms in applications. System administrators can use this text to select products and configure them appropriately.

Appendices are provided to supplement the main text in each part where a topic demands a more detailed treatment.

2 Glossary of Terms and Acronyms

Definitions provided below are defined as used in this document. The same terms may be defined differently in other documents.

2.1 Glossary

Access authority	An entity responsible for monitoring and granting access privileges for other authorized entities.
Access control	Restricts access to resources only to privileged entities.
Accountability	A property that ensures that the actions of an entity may be traced uniquely to that entity.
Approved	FIPS-Approved and/or NIST-recommended. An algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, or 2) adopted in a FIPS or NIST Recommendation and specified either in an appendix to the FIPS or NIST Recommendation, or in a document referenced by the FIPS or NIST Recommendation.
Archive	See Key management archive.
Association	A relationship for a particular purpose. For example, a key is associated with the application or process for which it will be used.
Asymmetric key algorithm	See Public key cryptographic algorithm.
Authentication	A process that establishes the origin of information, or determines an entity's identity.
Authentication code	A cryptographic checksum based on an Approved security function (also known as a Message Authentication Code).
Authorization	Access privileges granted to an entity; conveys an "official" sanction to perform a security function or activity.
Availability	Timely, reliable access to information by authorized entities.
Backup	A copy of information to facilitate recovery, if necessary.
Certificate	See public key certificate.
Certification authority	The entity in a Public Key Infrastructure (PKI) that is responsible for issuing certificates, and exacting compliance to a PKI policy.
Ciphertext	Data in its encrypted form.
Collision	Two or more distinct inputs produce the same output. Also see hash function.

Compromise	The unauthorized disclosure, modification, substitution or use of sensitive data (e.g., keying material and other security related information).
Confidentiality	The property that sensitive information is not disclosed to unauthorized entities.
Contingency plan	A plan that is maintained for disaster response, backup operations, and post-disaster recovery to ensure the availability of critical resources and to facilitate the continuity of operations in an emergency situation.
Cryptanalysis	1. Operations performed in defeating cryptographic protection without an initial knowledge of the key employed in providing the protection. 2. The study of mathematical techniques for attempting to defeat cryptographic techniques and information system security. This includes the process of looking for errors or weaknesses in the implementation of an algorithm or of the algorithm itself.
Cryptographic key (key)	A parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce or reverse the operation, while an entity without knowledge of the key cannot. Examples include: 1. the transformation of plaintext data into ciphertext data, 2. the transformation of ciphertext data into plaintext data, 3. the computation of a digital signature from data, 4. the verification of a digital signature, 5. the computation of an authentication code from data, 6. the verification of an authentication code from data and a received authentication code, 7. the computation of a shared secret that is used to derive keying material.
Cryptographic key component (key component)	One of at least two parameters that have the same format as a cryptographic key; parameters are combined in an Approved security function to form a plaintext cryptographic key before use.
Cryptographic module	The set of hardware, software, and/or firmware that implements Approved security functions (including cryptographic algorithms and key generation) and is contained within the cryptographic boundary.
Cryptomodule	See cryptographic module.
Cryptoperiod	The time span during which a specific key is authorized for use or in which the keys for a given system or application may remain in effect.
Data key, Data encrypting key	A cryptographic key that is used to cryptographically protect data (e.g., encrypt, decrypt, authenticate).

Data integrity	A property whereby data has not been altered in an unauthorized manner since it was created, transmitted or stored. In this guideline, the statement that a cryptographic algorithm "provides data integrity" means that the algorithm is used to detect unauthorized alterations.
Decryption	The process of changing ciphertext into plaintext using a cryptographic algorithm and key.
Digital signature	The result of a cryptographic transformation of data that, when properly implemented, provides the services of: 1. origin authentication 2. data integrity, and 3. signer non-repudiation.
Distribution	See key distribution.
Encrypted key	A cryptographic key that has been encrypted using an Approved security function with a key encrypting key in order to disguise the value of the underlying plaintext key.
Encryption	The process of changing plaintext into ciphertext using a cryptographic algorithm and key.
Entity	An individual (person), organization, device or process.
Ephemeral keys	Short-lived cryptographic keys that are statistically unique to each execution of a key establishment process and meets other requirements of the key type (e.g., unique to each message or session).
Hash-based message authentication code (HMAC)	A message authentication code that uses an Approved keyed hash function.
Hash function	A function that maps a bit string of arbitrary length to a fixed length bit string. Approved hash functions satisfy the following properties: 1. (One-way) It is computationally infeasible to find any input that maps to any pre-specified output, and 2. (Collision free) It is computationally infeasible to find any two distinct inputs that map to the same output.
Hash value	The result of applying a hash function to information.
Initialization vector (IV)	A vector used in defining the starting point of an encryption process within a cryptographic algorithm.

Integrity	The property that sensitive data has not been modified or deleted in an unauthorized and undetected manner. In this guideline, the stateme nt that a cryptographic algorithm "provides integrity" means that the algorithm is used to detect unauthorized modifications or deletions.
Key agreement	A key establishment protocol whose resultant keying material is a function of information contributed by two or more participants, so that no party can predetermine the value of the keying material.
Key component	See cryptographic key component.
Key confirmation	A method of determining that an entity has the correct keying material.
Key de-registration	A stage in the lifecycle of keying material; the removal of all records of keying material that was registered by a registration authority.
Key distribution	The transport of a key and other keying material from an entity that either owns the key or generates the key to another entity that is intended to use the key.
Key encrypting key	A cryptographic key that is used for the encryption or decryption of other keys.
Key establishment	A stage in the lifecycle of keying material; the process by which cryptographic keys are securely distributed among cryptographic modules using manual transport methods (e.g., key loaders), automated methods (e.g., key transport and/or key agreement protocols), or a combination of automated and manual methods (consists of key transport plus key agreement).
Keying material installation	A stage in the lifecycle of keying material; the installation of keying material for operational use.
Key management	The activities involving the handling of cryptographic keys and other related security parameters (e.g., IVs and passwords) during the entire life cycle of the keys, including their generation, storage, establishment, entry and output, and destruction.
Key management archive	A stage in the lifecycle of keying material; a repository containing keying material of historical interest.
Key Management Policy	The Key Management Policy is a high-level statement of organizational key management policies that identifies high-level structure, responsibilities, governing standards and guidelines, organizational dependencies and other relationships, and security policies.
Key Management Practices Statement	The Key Management Practices Statement is a document or set of documentation that describes in detail the organizational structure, responsible roles, and organization rules for the functions identified in the Key Management Policy.

Key pair	A public key and its corresponding private key; a key pair is used with a public key algorithm.
Key recovery	A stage in the lifecycle of keying material; mechanisms and processes that allow authorized entities to retrieve keying material from key backup or archive.
Key registration	A stage in the lifecycle of keying material; the process of officially recording the keying material by a registration authority.
Key revocation	A stage in the lifecycle of keying material; a process whereby a notice is made available to affected entities that keying material **should** be removed from operational use prior to the end of the established cryptoperiod of that keying material.
Key share	[Will not be addressed at this time.]
Key transport	Secure transport of cryptographic keys from one cryptographic module to another module. When used in conjunction with a public key (asymmetric) algorithm, keying material is encrypted using a public key and subsequently decrypted using a private key. When used in conjunction with a symmetric algorithm, key transport is known as key wrapping.
Key update	A stage in the lifecycle of keying material; alternate storage for operational keying material during its cryptoperiod.
Key wrapping	Encrypting a symmetric or asymmetric key using a symmetric key (the key encrypting key). A key used for key wrapping is known as a key encrypting key.
Key wrapping key	A symmetric key encrypting key.
Keying material	The data (e.g., keys and IVs) necessary to establish and maintain cryptographic keying relationships.
Label	Information that either identifies an associated parameter or provides information regarding the parameter's proper protection and use.
Manual key transport	A non-electronic means of transporting cryptographic keys by physically moving a device, document or person containing or possessing the key or a key component.
Message authentication code (MAC)	Data that is associated with authenticated information that allows an entity to verify the integrity of the information.
Nonce	A time-varying value that has, at most, a negligible chance of repeating. For example, a nonce could be a random value that is generated anew for each instance of a nonce, a timestamp, a sequence number, or some combination of these.

Non-repudiation	A service that is used to provide proof of the integrity and origin of data in such a way that the integrity and origin can be verified by a third party as having originated from a specific entity in possession of the private key of the nominal originator.
Operational storage	A stage in the lifecycle of keying material; the normal storage of operational keying material during its cryptoperiod.
Operational use	A stage in the lifecycle of keying material; a stage whereby keying material is used for standard cryptographic purposes.
Owner	For an asymmetric key pair, the entity that owns the private key, whether that entity generated the key pair, or a trusted party generated the key pair for the entity.
Originator usage period	The period of time in the cryptoperiod of a symmetric key during which cryptographic protection may be applied to data.
Password	A string of characters (letters, numbers and other symbols) that are used to authenticate an identity or to verify access authorization. Guidance on the creation and use of passwords is provided in SP-XXX.
Period of protection	The period of time during which keying material needs to be protected.
Plaintext	Intelligible data that has meaning and can be understood without the application of decryption.
Private key	A cryptographic key, used with a public key cryptographic algorithm, that is uniquely associated with an entity and is not made public. In an asymmetric (public) cryptosystem, the private key is associated with a public key. The private key is known only by the owner of the key pair and is used to: 1. Compute the corresponding public key, 2. Compute a digital signature that may be verified by the corresponding public key, 3. Decrypt data that was encrypted by the corresponding public key, or 4. Compute a piece of common shared data, together with other information.
Proof-of-possession (POP)	A verification process whereby it is proven that the owner of a key pair actually has the private key associated with the public key.
Pseudorandom number generator (PRNG)	An algorithm that produces a sequence of bits that are uniquely determined from an initial value called a seed. The output of the PRNG “appears” to be random, i.e., the output is statistically indistinguishable from random values. A cryptographic PRNG has the additional property that the output is unpredictable, given that the seed is not known.

Public key	A cryptographic key used with a public key cryptographic algorithm that is uniquely associated with an entity and that may be made public. In an asymmetric (public) cryptosystem, the public key is associated with a private key. The public key may be known by anyone and is used to: 1. Verify a digital signature that is signed by the corresponding private key, 2. Encrypt data that can be decrypted by the corresponding private key, or 3. Compute a piece of shared data.
Public key certificate	A set of data that uniquely identifies an entity, contains the entity's public key and possibly other information, and is digitally signed by a trusted party, thereby binding the public key to the entity. Additional information in the certificate could specify how the key is used and its cryptoperiod.
Public key (asymmetric) cryptographic algorithm	A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that determining the private key from the public key is computationally infeasible.
Public Key Infrastructure (PKI)	A framework that is established to issue, maintain and revoke public key certificates.
Public seed	A starting value for a pseudorandom number generator. The value produced by the random number generator may be made public. The public seed is often called a “salt”.
Random number generator (RNG)	A process used to generate an unpredictable series of numbers. Each individual value is called random if each of the values in the total population of values has an equal probability of being selected.
Recipient usage period	The period of time during the cryptoperiod of a symmetric key when protected information is processed. The recipient usage period of the key is usually identical to the cryptoperiod of that key.
RSA	The public key algorithm that was developed by Rivest, Shamir, and Adleman.
Salt	A value that may be publicly known and is sometimes used in cryptographic processes.
Secret key	A cryptographic key that is used with a secret key (symmetric) cryptographic algorithm, that is uniquely associated with one or more entities and is not be made public. The use of the term “secret” in this context does not imply a classification level, but rather implies the need to protect the key from disclosure.

Secret seed	A secret value that used to initialize a pseudorandom number generator. The resulting value from the random number generator remains secret or private.
Secure communication protocol	A communication protocol that provides the appropriate confidentiality, authentication and content integrity protection.
Security domain	A system or subsystem that is under the authority of a single trusted authority. Security domains may be organized (e.g., hierarchically) to form larger domains.
Security services	Mechanisms used to provide confidentiality, data integrity, authentication or non-repudiation of information.
Self-signed certificate	A public key certificate whose digital signature may be verified by the public key contained within the certificate. The signature on a self-signed certificate protects the integrity of the data, but does not guarantee authenticity of the information. The trust of self-signed certificates is based on the secure procedures used to distribute them.
Shall	This term is used to indicate a requirement of a Federal Information processing Standard (FIPS) or a requirement that must be fulfilled to claim conformance to this Recommendation. Note that **shall** may be coupled with **not** to become **shall not**.
Shared secret	A value that is generated during a key agreement process; the shared secret is typically used to derive keying material for a symmetric key algorithm.
Should	This term is used to indicate a very important requirement. While the "requirement" is not stated in a FIPS, ignoring the requirement could result in undesirable results. Note that **should** may be coupled with **not** to become **should not**.
Signature generation	Uses a digital signature algorithm and a private key to generate a digital signature on data.
Signature verification	Uses a digital signature algorithm and a public key to verify a digital signature.
Split knowledge	A procedure whereby a cryptographic key is handled as multiple key components from the time that the key or the separate key components are generated until the key components are combined for use. Each key component provides no knowledge of the ultimate key. The key may be created and then split into the key components, or may be created as separate key components. The key components are output from the generating cryptographic module(s) to separate entities for individual handling, and subsequently input separately into the intended cryptographic module and combined to form the ultimate key. Note: A suitable combination function is not provided by simple concatenation; e.g., it is not acceptable to form an 80 bit key by concatenating two 40-bit key components.

Static keys	Static keys are relatively long-lived and are common to a number of executions of a given algorithm.
Statistically unique	For the generation of *n*-bit quantities, the probability of two values repeating is less than or equal to the probability of two *n*-bit random quantities repeating.
Symmetric key	A single cryptographic key that is used with a secret (symmetric) key algorithm.
Symmetric key algorithm	A cryptographic algorithm that uses one shared key, a secret key.
System initialization	A stage in the lifecycle of keying material; setting up and configuring a system for secure operation.
Threat	Any circumstance or event with the potential to adversely impact a system through unauthorized access, destruction, disclosure, modification of data or denial of service.
Trust anchor	A public key and the name of a certification authority that is used to validate the first certificate in a sequence of certificates. The trust anchor public key is used to verify the signature on a certificate issued by a trust anchor certification authority. The security of the validation process depends upon the authenticity and integrity of the trust anchor. Trust anchors are often distributed as self-signed certificates.
Unauthorized disclosure	An event involving the exposure of information to entities not authorized access to the information.
User initialization	A stage in the lifecycle of keying material; the process whereby a user initializes its cryptographic application (e.g., installing and initializing software and hardware).
User registration	A stage in the lifecycle of keying material; a process whereby an entity becomes a member of a security domain.
Work	The expected time to break a cipher with a given resource. For example, 12 MIPS years would be the amount of work that one computer, with the capability of processing a Million Instructions Per Second, could do in 12 years. The same amount of work could be done by 12 such computer in one year.
X.509 certificate	The ISO/ITU-T X.509 standard defined two types of certificates – the X.509 public key certificate, and the X.509 attribute certificate. Most commonly (including this document), an X.509 certificate refers to the X.509 public key certificate.
X.509 public key certificate	The public key for a user (or device) and a name for the user (or device), together with some other information, rendered unforgeable by the digital signature of the certification authority that issued the certificate, encoded in the format defined in the ISO/ITU-T X.509 standard.

2.2 Acronyms

The following abbreviations and acronyms are used in this standard:

2DES	Two key Triple DES
3DES	Three key Triple DES
AES	Advanced Encryption Standard specified in FIPS 197.
ANSI	American National Standards Institute
CA	Certification Authority
CKL	Compromised Key List
CRC	Cyclic Redundancy Check
CRL	Certificate Revocation List
DEK	Data Encryption Key
DSA	Digital Signature Algorithm specified in FIPS 186-2.
ECDSA	Elliptic Curve Digital Signature Algorithm specified in ANSI X9.62.
FIPS	Federal Information Processing Standard.
HMAC	Keyed-Hash Message Authentication Code specified in FIPS 198.
IV	Initialization Vector.
KEK	Key Encrypting Key
LAN	Local Area Network
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OMB	Office of Management and Budget
PKI	Public Key Infrastructure
POP	Proof-of-Possession
PRNG	Pseudorandom Number Generator
RA	Registration Authority
RNG	Random Number Generator
RSA	Rivest, Shamir, Adelman (an algorithm)
TDES	Triple Data Encryption Standard; Triple DES
TLS	Transport Layer Security

3. Security Services

Cryptography may be used to perform several basic security services: confidentiality, data integrity, authentication, authorization and non-repudiation. These services may also be required to protect cryptographic keying material. In addition, there are other cryptographic and non-cryptographic mechanisms that are used to support these security services. In general, a single cryptographic process often provides more than one service (e.g., the use of digital signatures can provide integrity, authentication and non-repudiation).

Appendix A.1 provides a discussion of cryptographic and non-cryptographic methods for providing data integrity and authentication services in communication protocols.

3.1 Confidentiality

Confidentiality is the property whereby information is not disclosed to unauthorized parties. Secrecy and privacy are terms that are often used synonymously with confidentiality.

Confidentiality that is achieved using cryptography makes use of encryption to render the information unintelligible except by authorized entities. The information may become intelligible again by using decryption.

3.2 Data Integrity

Data integrity is the property whereby data is not modified in an unauthorized manner; this includes the insertion, deletion and substitution of data. Cryptographic algorithms can be used to detect errors in the information (i.e., detect that the information has been modified). Absolute protection against modification is not possible. However, well-designed detection mechanisms can be used to support the maintenance of data integrity.

Cryptographic mechanisms, such as message authentication codes or digital signatures, can be used to detect (with a high probability) both accidental modifications (e.g., modifications that sometimes occur during noisy transmissions or by hardware memory failures), and deliberate modifications by an adversary with a very high probability. Non-cryptographic mechanisms are also often used to detect accidental modifications, but cannot be relied upon to detect deliberate modifications. A more detailed treatment of this subject is provided in Appendix A.1.

In this guideline, the statement that a cryptographic algorithm "provides data integrity" means that the algorithm is used to detect unauthorized alterations.

3.3 Authentication

Authentication is a service that is used to establish the origin of information. That is, authentication services verify the identity of the user or system that created information (e.g., a transaction or message). This service supports the receiver in security relevant decisions, such as “Is the sender an authorized user of this system?” or “Is the sender permitted to read sensitive information?” Several cryptographic mechanisms may be used to provide authentication services. Most commonly, authentication is provided by digital signatures or message authentication codes; some key agreement techniques also provide authentication.

3.4 Authorization

Authorization is concerned with providing an official sanction or permission to perform a security function or activity. Normally, authorization is granted following a process of authentication. A non-cryptographic analog of the interaction between authentication and authorization is the examination of an individual's credentials to establish their identity (authentication); upon proving identity, the individual is then provided with the key or password that will allow access to some resource, such as a locked room (authorization).

3.5 Non-repudiation

Non-repudiation is a service that is used to provide proof of the integrity and origin of data in such a way that the integrity and origin can be verified by a third party. This service prevents an entity from successfully denying involvement in a previous action. Non-repudiation is provided cryptographically by the use of a digital signature that is calculated by a private key known only by the entity that computes the digital signature.

3.6 Support Services

Cryptographic security services often require supporting services. For example, cryptographic services often require the use of key establishment and random number generation services.

3.7 Combining Services

In many applications a combination of cryptographic services (confidentiality, data integrity, authentication, authorization, and non-repudiation) is desired. Designers of secure systems often begin by considering which security services are needed to protect the information contained within and processed by the system. After these services have been determined, the designer then considers what mechanisms will best provide these services. Not all mechanisms are cryptographic in nature. For example, physical security may be used to protect the confidentiality of certain types of data, and identification badges or biometric identification devices may be used for entity authentication. However, cryptographic mechanisms consisting of algorithms, keys, and other keying material often provide the most cost-effective means of protecting the security of information. This is particularly true in applications where the information would otherwise be exposed to unauthorized entities.

When properly implemented, some cryptographic algorithms provide multiple services. The following examples illustrate this case:

1. A message authentication code (Section 4.2.3) can provide authentication as well as data integrity if the symmetric keys are unique to each pair of users.
2. A digital signature algorithm (Section 4.2.4) can provide authentication and data integrity, as well as non-repudiation.
3. Certain modes of encryption can provide confidentiality, data integrity, and authentication when properly implemented. These modes **should** be specifically designed to provide these services.

However, it is often the case that different algorithms must be employed in order to provide all the desired services.

For example, suppose that the secure exchange of information between pairs of Internet entities is needed. Some of the exchanged information requires just integrity protection, while other information requires both integrity and confidentiality protection. It is also a requirement that each entity that participates in an information exchange knows the identity of the other entity.

The designers of this system decided that a Public Key Infrastructure (PKI) would be established and that each entity wishing to communicate securely would be required to physically authenticate their identity at a Registration Authority (RA). This authentication would require the presentation of proper credentials such as a driver's license, passport, or birth certificate. The authenticated individuals would then generate a static public key pair in a smart card that would be used for key agreement (i.e., key establishment). The public static key agreement key of each net member would be transferred from the smart card to the RA where it is incorporated with the user identity and other information into a digitally signed message for transmission to a Certificate Authority (CA). The CA would then compose the user's public key certificate by signing the public key of the user and the user's identity along with other information. This certificate is returned to the public key owner so that it may be used in conjunction with the private key (under the sole control of the owner) for entity authentication and key establishment purposes.

Any two entities wishing to communicate may exchange public key certificates containing that are checked by verifying the CA signature on the certificate (using the CA public key). The public static key agreement key of each entity and each entity's own private static key agreement key is then used in a key establishment scheme to produce a secret value shared between the two entities. The shared secret may then be used to derive one or more shared symmetric keys. If the mode of the symmetric encryption algorithm is designed to support all the desired services, then only one shared key is necessary. Otherwise multiple shared keys and algorithms could be used. One of the shared keys could be used to encrypt for confidentiality, while another key could be used for integrity and authentication. The receiver of the data protected by the key(s) would have assurance that the data came from the other entity indicated by the public key certificate, that the data remained confidential, and that the integrity of the data was preserved.

Alternatively, if confidentiality were not required, integrity protection, entity authentication, and non-repudiation could be attained by establishing a signature key pair and corresponding certificate for each entity. The private signature key of the sender would be used to sign the data, and the sender's public signature verification key would be used by the receiver to verify the signature. In this case a single algorithm provides all three services.

The above example provides a basic sketch of how cryptographic algorithms may be used to support multiple security services. However, it can be easily seen that the security of such a system depends on many factors including:

a. The strength of the entity's credentials (e.g., driver's license, passport, or birth certificate) and authentication mechanism,

b. The strength of the cryptographic algorithms used,

c. The degree of trust placed in the RA and the CA,

d. The strength of the key establishment protocols, and

The care of the users to protect their smart cards (and thus, their keys) from unauthorized use.

Therefore the design of a security system that provides desired security services by making use of cryptographic algorithms and sound key management techniques requires a high degree of skill and expertise.

4. Cryptographic Algorithms

FIPS-approved or NIST-recommended cryptographic algorithms **shall** be used whenever cryptographic services are required. These Approved algorithms have received an intensive security analysis prior to their approval and continue to be examined to determine that the algorithms provide adequate security. Most cryptographic algorithms require cryptographic keys or other keying material. In some cases, an algorithm may be strengthened by the use of larger keys. This guideline advises the users of cryptographic mechanisms on the approved choices of algorithms and key sizes.

This section describes the Approved cryptographic algorithms that provide security services such as confidentiality, data integrity, authentication, authorization, non-repudiation, and algorithms that support these services.

4.1 Classes of Cryptographic Algorithms

There are three basic classes of Approved cryptographic algorithms, hash algorithms, symmetric key algorithms and asymmetric key algorithms. The classes are defined by the number of cryptographic keys that are used in conjunction with the algorithm.

Cryptographic hash algorithms (i.e., hash functions) require no keys. Hash algorithms generate a small message digest (hash value) from a (possibly) large message and are used as components in many cryptographic processes, such as Message Authentication Codes (MACs) (Section 4.2.3), digital signatures (Section 4.2.4), key establishment (Section 4.2.5), and random number generation (Section 4.2.6)).

Symmetric key algorithms (sometimes known as secret key algorithms) use a single key to transform data. Symmetric keys are often known by more than one entity; however, the key **should not** be known by entities that are not authorized access to the data protected by that algorithm and key. Symmetric key algorithms are used:

1. To provide data confidentiality (Section 4.2.2) - the same key is used to encrypt and decrypt data.
2. To provide authentication and integrity services (Section 4.2.3) in the form of Message Authentication Codes (MACs) - the same key is used to generate the MAC and to validate it. MACs normally employ either a symmetric key encryption algorithm or a cryptographic hash function as their cryptographic primitive.
3. As part of the key establishment process (Section 4.2.5).
4. To generate pseudorandom numbers (Section 4.2.6).

Asymmetric key algorithms, commonly known as public key algorithms, use two related keys (i.e., a key pair) to perform their functions: a public key and a private key. The public key may be known by anyone; the private key **should** be under the sole control of the entity that "owns" the key pair. Even though the public and private keys of a key pair are related, knowledge of the public key does not reveal the private key. Public key algorithms are commonly used in the computation of digital signatures (Section 4.2.4) and in the establishment of cryptographic keying material (Section 4.2.5).

4.2 Cryptographic Algorithm Functionality

Security services are fulfilled using a number of different algorithms. In many cases, the same algorithm may be used to provide multiple services.

4.2.1 Hash Function

Hash functions are used with other algorithms to provide a number of security services. A hash function is used in conjunction with a digital signature algorithm to compute a digital signature (see [FIPS 186-2]). A hash function may be used as part of a random number generator. A hash function is used with a key as part of its input to provide a Keyed-Hash Message Authentication Code (HMAC) (see Section 4.2.3.2 and [FIPS 198]). Approved hash functions are defined in [FIPS 180-2].

A hash function takes an input of arbitrary length and outputs a fixed size value. Common names for the output of a hash function include hash value, hash, message digest, and digital fingerprint. The maximum number of input and output bits is determined by the design of the hash function. All Approved hash functions are cryptographic hash functions. With a well-designed hash function, it is not feasible to find two messages that produce the same hash value.

Four hash functions are approved for Federal Government use and are defined in [FIPS 180-2]: SHA-1, SHA-256, SHA-384 and SHA-512. The hash size produced by SHA-1 is 160 bits, 256 bits for SHA-256, 384 bits for SHA-384, and 512 bits for SHA-512.

4.2.2 Symmetric Key Algorithms used for Encryption and Decryption

Encryption is used to provide confidentiality for data. The data to be protected is called plaintext when in its original form. Encryption transforms the data into ciphertext. Ciphertext can be transformed back into plaintext using decryption. The approved algorithms for encryption/decryption are symmetric key algorithms: AES and TDES. Each of these algorithms operates on blocks (chunks) of data during an encryption or decryption operation. For this reason, these algorithms are commonly referred to as block cipher algorithms.

4.2.2.1 Advanced Encryption Standard (AES)

The AES algorithm is specified in [FIPS 197]. AES encrypts and decrypts data in 128-bit blocks, using either 128, 192 or 256 bit keys. The nomenclature for AES for the different key sizes is AES-x, where x is the key size. All three key sizes are considered adequate for Federal Government applications.

4.2.2.2 Triple DES (TDES)

Triple DES is defined in [ANSI X9.52] and has been adopted in [FIPS 46-3]. TDES encrypts and decrypts data in 64-bit blocks, using three 56-bit keys. Federal applications **should** use three distinct keys. [ANSI X9.52] specifies seven modes of operation that are used with TDES for encryption.

4.2.2.3 Modes of Operation

With a block cipher algorithm, the same plaintext block will always encrypt to the same ciphertext block whenever the same key is used. If the multiple blocks in a typical message were to be encrypted separately, an adversary could easily substitute individual blocks, possibly without detection. Furthermore, certain kinds of data patterns in the plaintext, such as repeated blocks, would be apparent in the ciphertext.

Cryptographic modes of operation have been defined to alleviate this problem by combining the basic cryptographic algorithm with some sort of feedback of the information derived from the cryptographic operation. The NIST Recommendation for Block Cipher Modes of Operation [SP 800-38A] defines modes of operation for the encryption and decryption of data using block cipher algorithms such as AES and TDES. [ANSI X9.52], which was adopted by [FIPS 46-3], defines three additional TDES modes that pipeline or interleave the cryptographic operations to improve the efficiency of TDES.

4.2.3 Message Authentication Codes (MACs)

Message Authentication Codes (MACs) provide data authentication and integrity. A MAC is a cryptographic checksum on the data that is used to provide assurance that the data has not changed and that the MAC was computed by the expected entity. Although message integrity is often provided using non-cryptographic techniques known as error detection codes, these codes can be altered by an adversary to effect an action to the adversary's benefit. The use of an Approved cryptographic mechanism, such as a MAC, can alleviate this problem. In addition, the MAC can provide a recipient with assurance as to the identity of the originator of any data for which a MAC is provided.

The computation of a MAC requires the use of a secret key that is known only by the party that generates the MAC and by the intended recipient(s) of the MAC, and the data on which the MAC is calculated. Two types of algorithms for computing a MAC have been approved: MAC algorithms that are based on block cipher algorithms, and MAC algorithms that are based on hash functions.

4.2.3.1 MACs Using Block Cipher Algorithms

[SP 800-38B] defines a mode to compute a MAC using Approved block cipher algorithms such as AES and TDES. The key and block size used to compute the MAC depends on the algorithm used.

4.2.3.2 MACs Using Hash Functions

[FIPS 198] specifies the computation of a MAC using an Approved hash function. The algorithm requires a single pass through the entire data. A variety of key sizes are allowed for HMAC; the choice of key size depends on the amount of security to be provided to the data and the hash function used. See Section 8.8 for guidance in the selection of key sizes.

4.2.4 Digital Signature Algorithms

Digital signatures are used to provide authentication, integrity and non-repudiation. Digital signatures are used in conjunction with hash algorithms and are computed on data of any length (up to a limit that is determined by the hash algorithm). [FIPS 186-2] specifies algorithms that are approved for the computation of digital signatures; this standard defines the Digital Signature Algorithm (DSA) and adopts two algorithms specified in ANSI standards: [ANSI X9.31] (RSA and Rabin-Williams) and [ANSI X9.62] (the Elliptic Curve Digital Signature Algorithm - ECDSA). [A revision for FIPS 186-2 is planned. At that time, this section will be revised.]

4.2.4.1 DSA

The Digital Signature Algorithm (DSA) is specified in [FIPS 186-2] and will be revised as FIPS 186-3 in the near future. The revised standard will define this algorithm for specific key sizes[1]: 1024, 2048, 3072, 8192 and 15,360 bits. The revised DSA will produce digital signatures[2] of 320, 448, 512, 768 or 1024 bits. Older systems (legacy systems) used smaller key sizes. While it may be appropriate to continue to verify and honor signatures created using these smaller key sizes[3], new signatures **shall not** be created using these key sizes.

4.2.4.2 RSA

The RSA algorithm, as specified in [ANSI X9.31] was adopted for the computation of digital signatures in FIPS 186-2. The RSA algorithm is also used for digital signatures in [PKCS #1] (version 1.5 and higher) and will be adopted in [FIPS 186-3]. FIPS 186-3 will specify methods for generating RSA key pairs for several key sizes for both ANSI X9.31 and PKCS #1 implementations. Older systems (legacy systems) used smaller key sizes. While it may be appropriate to continue to verify and honor signatures created using these smaller key sizes[4], new signatures **shall not** be created using these key sizes.

4.2.4.3 ECDSA

The Elliptic Curve Digital Signature Algorithm (ECDSA), as specified in [ANSI X9.62], has been adopted for the computation of digital signatures in [FIPS 186-2]. [ANSI X9.62] specifies a minimum key size[5] of 160 bits. ECDSA produces digital signatures that are twice the length of the key size. Recommended elliptic curves are provided in [FIPS 186-2].

4.2.5 Key Establishment Algorithms

Key establishment algorithms are used to set up keys to be used between communicating entities. Two types of key establishment are defined: key transport and key agreement. Approved key establishment schemes are provided in [SP 800-56].

Key transport is the distribution of a key (and other keying material) from one entity to another entity. The keying material is usually encrypted by the sending entity and decrypted by the receiving entity(ies). If a symmetric algorithm (e.g., AES) is used to encrypt the keying material to be distributed, the sending and receiving entities need to know the symmetric key wrapping key (i.e., key encrypting key); in this case, the key transport algorithm is commonly known as a key wrapping algorithm. If a public key algorithm is used to distribute the keying material, a key pair is used as the key encrypting key; in this case, the sending entity encrypts the keying

[1] For DSA, the key size is considered to be the size of the modulus p. Another value, q, is also important when defining the security afforded by DSA.

[2] The length of the digital signature is twice the size of q (see the previous footnote).

[3] This may be appropriate if it is possible to determine when the digital signature was created (e.g., by the use of a trusted time stamping process). The decision should not depend solely on the cryptography used.

[4] This may be appropriate if it is possible to determine when the digital signature was created (e.g., by the use of a trusted time stamping process). The decision should not depend solely on the cryptography used.

[5] For elliptic curves, the key size is the length f of the order n of the base point G of the chosen elliptic curve.

material using the receiving entity's public key; the receiving entity decrypts the received keying material using the associated private key.

Key agreement is the participation by both entities (i.e., the sending and receiving entities) in the creation of shared keying material. This may be accomplished using either asymmetric (public key) or symmetric key techniques. If an asymmetric algorithm is used, each entity has either a static key pair or an ephemeral key pair or both. If a symmetric algorithm is used, each entity shares the same symmetric key wrapping key.

[SP 800-56] adopts selected key establishment schemes defined in ANSI standards that use public key algorithms: [ANSI X9.42], [ANSI X9.44], and [ANSI X9.63]. [ANSI X9.42] specifies key agreement schemes, and [ANSI X9.44] and [ANSI X9.63] specify both key agreement and key transport schemes. [SP 800-56] also specifies techniques for key wrapping.

4.2.5.1 Discrete Log Key Agreement Schemes Using Finite Field Arithmetic

Key agreement schemes based on the intractability of the discrete logarithm problem and using finite field arithmetic have been adopted in [SP 800-56] from ANSI [X9.42]. [SP 800-56] has adopted seven of the eight schemes defined in [ANSI X9.42]. Each scheme provides a different configuration of required key pairs that may be used, depending on the requirements of a communication situation.

4.2.5.2 Discrete Log Key Agreement Schemes Using Elliptic Curve Arithmetic

Key agreement schemes based on the intractability of the discrete logarithm problem and using elliptic curve arithmetic have been adopted in [SP 800-56] from [ANSI X9.63]. Seven of the eleven key agreement schemes have been adopted. Each scheme provides a different configuration of required key pairs that may be used, depending on the requirements of a communication situation.

4.2.5.3 RSA Key Transport

RSA key agreement and key transport schemes will be adopted from [ANSI X9.44]. [Further text to be developed, depending on the contents of SP 800-56.]

4.2.5.4 Key Wrapping

Key wrapping is the encryption of a key by a key encryption key using a symmetric algorithm (e.g., an AES key is encrypted by an AES key encrypting key). [Further text will be added as key wrapping is developed further.]

4.2.6 Random Number Generation

Random number generators (RNGs) are required for the generation of keying material (e.g., keys and IVs). Two classes of RNGs are defined: deterministic and non-deterministic. Deterministic RNGs, often called pseudorandom number generators, use cryptographic algorithms and the associated keying material to generate random numbers; non-deterministic RNGs, often called true RNGs, produce output that is dependent on some unpredictable physical source that is outside human control.

Pseudorandom number generation (PRNG) algorithms defined in [FIPS 186-2] (including those defined in [ANSI X9.31] and [ANSI X9.62][6]) use hash functions to generate random numbers that are used for cryptographic applications (e.g., key or IV generation). PRNGs are initialized with a starting value, called a seed. The seed may be public or secret, depending on its use.

[6] As allowed in FIPS 186-2.

5. Protection Requirements for Cryptographic Information

This section defines the different types of keys that are used by the Approved algorithms and gives guidance on the types of protection for keying material. Cryptographic keying material is defined as the cryptographic key and associated information required to use the key. The specific information varies depending on the type of key. The cryptographic keying material must be protected in order for the security services to be "meaningful." Much of the protection needed may be provided by FIPS 140-2 Approved cryptographic module (cryptomodule); however, whenever the keying material exists external to a FIPS 140-2 cryptomodule, additional protection is required. The type of protection needed depends on the type of key and the security service for which the key is used.

5.1 Protection Requirements

Keying material **should** be (operationally) available as long as the associated cryptographic service is required. Some keys may need to be archived if required beyond the key's cryptoperiod.

Integrity protection **shall** be provided for all keying material.. Integrity protection can be provided by cryptographic integrity mechanisms (e.g. cryptographic checksums, cryptographic hashes, MACs, and signatures), non-cryptographic integrity mechanisms (e.g. CRCs, parity, etc.) (see Appendix A.1), or physical protection mechanisms. Guidance for the selection of appropriate integrity mechanisms is given in Sections 5.2.1.2 and 5.2.2.2.

The confidentiality of all symmetric and private keys **shall** be protected. Public keys generally do not require confidentiality protection. When the key exists internal to an Approved cryptomodule, confidentiality protection is provided by the cryptomodule in accordance with FIPS 140-2. When the key exists external to the cryptomodule, confidentiality protection **shall** be provided either by encryption (e.g., key wrapping) or by controlling access to the key via physical means (e.g. storing the keying material in a safe with limited access). The security and operational impact of specific confidentiality mechanisms varies. Guidance for the selection of appropriate confidentiality mechanisms is given in Sections 5.2.1.3 and 5.2.2.3.

Association protection **shall** be provided to ensure the availability of a cryptographic security service by ensuring that the correct keying material is used with the correct data in the correct application or equipment. Guidance for the selection of appropriate association protection is given in Sections 5.2.1.4 and 5.2.2.4.

Public key validation and domain parameter validation ensures that the required mathematical properties are present (see Section 8.3). Guidance for the selection of appropriate validation mechanisms is given in [SP 800-56]

The period of protection for cryptographic keys, associated key information, and cryptographic parameters (e.g. initialization vectors, seeds, etc.) depends on the type of key, the associated cryptographic service, and the length of time for which the cryptographic service is required. The period of protection is related to the cryptoperiod of the key (see Section 8.2), which defines the time period for which the key is used operationally.

5.1.1 Cryptographic Keys

Several different types of keys are defined. The keys are identified according to their classification as public, private or symmetric keys and as to their use. For public and private key agreement keys, their status as static or ephemeral keys is also specified.

1. *Private signature key*: Private signature keys are the private keys of asymmetric (public) key pairs that are used by public key algorithms to generate digital signatures with possible long-term implications. When properly handled, private signature keys can be used to provide authentication, integrity and non-repudiation.
2. *Public signature verification key*: A public signature verification key is the public key of an asymmetric (public) key pair that is used by a public key algorithm to verify digital signatures, either to authenticate a user's identity, to determine the integrity of the data, for non-repudiation, or a combination thereof.
3. *Symmetric authentication key*: Symmetric authentication keys are used with symmetric key algorithms to provide assurance of the integrity and source of messages, communication sessions, or stored data.
4. *Private authentication key*: A private authentication key is the private key of an asymmetric (public) key pair that is used with a public key algorithm to provide assurance as to the integrity of information, and the identity of the originating entity or the source of messages, communication sessions, or stored data.
5. *Public authentication key*: A public authentication key is the pub lic key of an asymmetric (public) key pair that is used with a public key algorithm to determine the integrity of information and to authenticate the identity of entities, or the source of messages, communication sessions, or stored data.
6. *Symmetric data encryption key*: These keys are used with symmetric key algorithms to apply confidentiality protection to information.
7. *Symmetric key wrapping key*: Symmetric key wrapping keys are used to encrypt other keys using symmetric key algorithms. Key wrapping keys are also known as key encryption keys.
8. *Random number generation key*: These keys are symmetric (secret) keys used with a symmetric algorithm or hash function to generate pseudorandom numbers.
9. *Symmetric master key*: A symmetric master key is used to derive other symmetric keys (e.g., data encryption keys, key wrapping keys, or authentication keys) using symmetric cryptographic methods.
10. *Private key transport key*: Private key transport keys are the private keys of asymmetric (public) key pairs that are used to decrypt keys that have been encrypted with the associated public key using a public key algorithm. Key transport keys are usually used to establish keys (e.g., key wrapping keys, data encryption keys or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors).
11. *Public key transport key*: Public key transport keys are the public keys of asymmetric (public) key pairs that are used to encrypt keys using a public key algorithm. These keys

are used to establish keys (e.g., key wrapping keys, data encryption keys or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors).

12. *Symmetric key agreement key*: These symmetric keys are used to establish keys (e.g., key wrapping keys, data encryption keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors) using a symmetric key agreement algorithm.
13. *Private static key agreement key*: Private static key agreement keys are the private keys of asymmetric (public) key pairs that are used to establish keys (e.g., key wrapping keys, data encryption keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors).
14. *Public static key agreement key*: Public static key agreement keys are the public keys of asymmetric (public) key pairs tha t are used to establish keys (e.g., key wrapping keys, data encryption keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors).
15. *Private ephemeral key agreement key*: Private ephemeral key agreement keys are the private keys of asymmetric (public) key pairs that are used only once to establish one or more keys (e.g., key wrapping keys, data encryption keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors).
16. *Public ephemeral key agreement key*: Public ephemeral key agreement keys are the public keys of asymmetric (public) key pairs that are used only once to establish one or more keys (e.g., key wrapping keys, data encryption keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors).
17. *Symmetric authorization key*: Symmetric authorization keys are used to provide privileges to an entity using a symmetric cryptographic method. The authorization key is known by the access authority and the entity seeking access to resources.
18. *Private authorization key*: A private authorization key is the private key of an asymmetric (public) key pair that is used to provide privileges to an entity.
19. *Public authorization key*: A public authorization key is the private key of an asymmetric (public) key pair that is used to verify privileges for an entity that knows the associated private authorization key.

Table 1 provides a summary of the protection requirements for keys during distribution and storage. Methods for providing the necessary protection are discussed in Section 5.2.

Guide to Table 1:

a. Column 1 (Key Type) identifies the key types.
b. Column 2 (Security Service) indicates the type of security service that is provided by the key in conjunction with a cryptographic technique.
c. Column 3 (Security Protection) indicates the type of protection required for the key (archive availability, integrity, confidentiality).
d. Column 4 (Association Protection) indicates the types of associations that are applicable for that key, such as associating the key with the usage or application, the authorized communications participants or other indicated information **shall** be protected.

e. Column 5 (Validity Assurance) indicates the keys for which assurance of validity needs to be obtained as defined in [SP 800-56], [FIPS 186-3] and this guideline. See Section 8.3 for further details.

f. Column 6 (Period of Protection) indicates the length of time that the key needs to be protected. Symmetric keys and private keys are destroyed at the end of their cryptoperiod; however, back-up and archival requirements will extend the period of protection beyond the cryptoperiod of the key.

Table 1: Protection requirements for cryptographic keys.

Key Type	Security Service	Security Protection	Association Protection	Validity Assurance	Period of Protection
Private signature key	Authentication; Integrity; Non-repudiation	Integrity; Confidentiality	Usage or application; Domain parameters; Public signature verification key		The cryptoperiod of the key and until the private key is destroyed
Public signature verification key	Authentication; Integrity; Non-repudiation	Archive; Integrity;	Usage or application; Key pair owner Domain parameters; Private signature key; Signed data	For association with signing private key	As long as signed data may need to be verified
Symmetric authentication key	Authentication; Integrity	Archive; Integrity; Confidentiality	Usage or application; Other authorized entities; Authenticated data		As long as the authenticated data may need to be authenticated and until the key is destroyed
Private authentication key	Authentication; Integrity	Integrity; Confidentiality	Usage or application; Public authentication key; Domain parameters		The cryptoperiod of the key and until the key is destroyed
Public authentication key	Authentication; Integrity	Archive; Integrity	Usage or application; Key pair owner; Authenticated data; Private authentication key; Domain parameters	For association with private key	As long as the authenticated data may need to be authenticated

Key Type	Security Service	Security Protection	Association Protection	Validity Assurance	Period of Protection
Symmetric data encryption key	Confidentiality	Archive; Integrity; Confidentiality	Usage or application; Other authorized entities; Encrypted data		The cryptoperiod of the key and the lifetime of the data and until the key is destroyed
Symmetric key wrapping key	Support	Archive; Integrity; Confidentiality	Usage or application; Other authorized entities; Encrypted keys		The cryptoperiod of the key and until the key is destroyed
Symmetric RNG key	Support	Integrity; Confidentiality	Usage or application		Until no longer needed to generate or reconstruct random numbers and until destroyed
Symmetric master key	Support	Archive; Integrity; Confidentiality	Usage or application; Other authorized entities; Derived keys		The cryptoperiod of the key, the lifetime of any keys derived using this key, and until the key is destroyed
Private key transport key	Support	Archive; Integrity; Confidentiality	Usage or application; Encrypted keys; Public key transport key; Domain parameters		As long as the transported key needs to be decrypted and until destroyed
Public key transport key	Support	Integrity	Usage or application; Key pair owner; Private key transport key; Domain parameters	Yes	The cryptoperiod of the key
Symmetric key agreement key	Support	Archive; Integrity; Confidentiality	Usage or application; Other authorized entities		The cryptoperiod of the key, until no longer needed to determine a key, and until destroyed
Private static key agreement key	Support	Archive; Integrity; Confidentiality	Usage or application; Domain parameters; Public static key agreement key		The cryptoperiod of the key, until no longer needed to determine a key, and until destroyed

Key Type	Security Service	Security Protection	Association Protection	Validity Assurance	Period of Protection

Public static key agreement key	Support	Archive;
		Integrity
			Usage or application;
			Key pair owner;
			Domain parameters;
			Private static key agreement key
				Yes	Until no longer needed to determine a key

Private ephemeral key agreement key	Support	Integrity;
		Confidentiality
			Usage or application;
			Public ephemeral key agreement key;
			Domain parameters;
					Until the key agreement process is complete; the ephemeral key is destroyed after the key agreement process is complete

Public ephemeral key agreement key	Support	Integrity[7]	Key pair owner;
			Private ephemeral key agreement key;
			Usage or application;
			Domain parameters
				Yes	Until the key agreement process is complete

Symmetric authorization keys	Authorization	Integrity; Confidentiality	Usage or application; Other authorized entities		The cryptoperiod of the key and until the key is destroyed

Private authorization key	Authorization	Integrity;
		Confidentiality
			Usage or application;
			Public authorization key;
			Domain parameters
					The cryptoperiod of the key and until the key is destroyed

Public authorization key	Authorization	Integrity	Usage or application;
			Key pair owner;
			Private authorization;
			Domain parameters
				Yes	The cryptoperiod of the key or
					As long as the authorization needs to be verified

[7] Although ephemeral keys are normally used during a very short period (e.g., during one communication session), they require integrity protection from the time that they are generated until the keys are no longer retained. This period of retention may be long enough for an adversary to substitute a key known to the adversary without detection unless integrity protection is provided.

5.1.2 Other Cryptographic or Related Information

Other information used in conjunction with cryptographic algorithms and keys also **should** be protected.

1. *Domain Parameters*: Domain parameters are used in conjunction with some public key algorithms to generate key pairs or to create digital signatures (i.e., DSA, and the Diffie-Hellman and MQV key agreement schemes specified in [SP 800-56]) and **shall** be protected in accordance with Table 2.
2. *Initialization Vectors*: Initialization vectors (IVs) are used by several modes of operation for encryption and decryption (see Section 4.2.2.3) and for the computation of MACs using block cipher algorithms (see Section 4.2.3.1) and **shall** be protected in accordance with Table 2.
3. *Shared Secrets:* Shared secrets are generated during a key establishment process as defined in [SP 800-56]. Shared secrets **shall not** exist outside the cryptographic boundary of the cryptomodule. If a FIPS 140-2 Approved cryptomodule is being used, then protection of the shared secrets is provided by the cryptomodule.
4. *Secret seeds*: Secret seeds are used in the generation of *pseudorandom* numbers that **shall** remain secret (e.g., used to generate keying material that must remain secret or private).
5. *Public seeds*: Public seeds are used in the generation of pseudorandom numbers that may be validated (e.g., elliptic curve domain parameters). A public seed is sometimes called a "salt".
6. *Other public information*: Public information (e.g., a nonce) is often used in the key establishment process.
7. *Intermediate Results*: The intermediate results of cryptographic operations using secret information **shall** be protected. In this case, intermediate results **shall not** exist outside the cryptographic boundary of the cryptomodule. If a FIPS 140-2 Approved cryptomodule is being used, then protection of the intermediate results is taken care of by the cryptomodule.
8. *Key control information*: Information related to the keying material (e.g., the identity, purpose, or a counter) **shall** be protected to ensure that the associated keying material can be correctly used.
9. *Random numbers*: The random numbers created by a random number generator **should** be protected when retained. When used directly as keying material, the random numbers **shall** be protected as discussed in Section 5.1.1.
10. *Passwords*: A password is used to acquire access to privileges. As such, it is used as an authentication mechanism.
11. *Audit information*: Audit information that contains key management events **shall** be protected.

Table 2 provides a summary of the protection requirements for this material during distribution and storage. Methods for providing the necessary protection are discussed in Section 5.2.

Guide to Table 2:

a. Column 1 (Cryptographic Information Type) identifies the type of cryptographic information.

b. Column 2 (Security Service) indicates the type of security service provided by the cryptographic information.

c. Column 3 (Security Protection) indicates the type of security protection for the cryptographic information.

d. Column 4 (Association Protection) indicates the relevant types of associations for each type of cryptographic information..

e. Column 5 (Validity Assurance) indicates the cryptographic information for which assurance **shall** obtained as defined in [SP 800-56] and in this guideline.

f. Column 6 (Period of Protection) indicates the length of time for the protection of the cryptographic information. This is generally similar to the period of protection for the associated keys. The cryptographic information is destroyed when no longer needed, unless specific back-up or archival requirements apply (see Sectio ns 7.3 and 8.7).

Table 2. Protection requirements for other cryptographic or related material.

Crypto. Information Type	Security Service	Security Protection	Association Protection	Validity Assurance Protection	Period of Protection
Domain parameters	Depends on key assoc. with the params.	Archive; Integrity	Usage or application; Private and public keys	Yes	Until no longer needed to generate keys, or verify signatures
Initialization vectors	Depends on algorithm	Archive; Integrity	Protected data		Until no longer needed to process the protected data
Shared secrets	Support	Integrity; Confidentiality			Until no longer needed to generate keying material, and the shared secret is destroyed
Secret seeds	Support	Confidentiality; Integrity	Usage or application		Until no longer needed to generate keying material and the shared secret is destroyed
Public seeds	Support	Archive; Integrity	User or application; Generated data		Until no longer needed to process generated data

Other public information	Support	Archive;
		Integrity;
			User or application;
			Other authorized entities;
			Data processed using the nonce
					Until no longer needed to process data using the public information

Intermediate results	Support	Confidentiality; Integrity	Usage or application		Until no longer needed and the intermediate results are destroyed
Key control information (e.g., IDs, purpose)	Support	Archive; Integrity	Key		Until the associated key is destroyed

Crypto. Information Type	Security Service	Security Protection	Association Protection	Validity Assurance Protection	Period of Protection
Random number	Support	Integrity; Confidentiality			Until no longer needed, and the random number is destroyed
Password	Authentication	Archive Integrity; Confidentiality	Usage or application; Owning entity		Until replaced or no longer needed to authenticate the entity
Audit information	Support	Archive; Integrity; Access authorization	Audited events; Key control information		Until no longer needed

5.2 Protection Mechanisms

During the lifetime of cryptographic information, the information is either "in transit" (e.g., is in the process of being manually or electronically distributed to the authorized communications participants for use by those entities) or is "at rest" (e.g., the information is in storage). In either case, the keying material **shall** be protected in accordance with Section 5.1. However, the choice of protection mechanisms may vary. Although several methods of protection are provided in the following subsections, not all methods provide equal security. The method **should** be carefully selected.

5.2.1 Protection Mechanisms for Cryptographic Information in Transit

Cryptographic information in transit may be keying material being distributed in order to obtain a cryptographic service (e.g., establish a key that will be used to provide confidentiality) (see Section 7.1.5), or cryptographic information being backed up or archived for possible use or recovery in the future (see Sections 7.2.2 and 7.3.1). This may be accomplished using "manual methods" (i.e., the information is in hard copy or on an electronic media, such as a CD-ROM), or using electronic communication protocols. For some protocols, the protections are provided by the protocol; in other cases, the protection for the keying material is be provided directly on the keying material. It is the responsibility of the originating entity to apply protection mechanisms, and the responsibility of the recipient to undo or check the mechanisms used.

5.2.1.1 Availability

Availability of cryptographic information in transit cannot be assured using cryptographic methods. Availability may be provided by retransmission.

5.2.1.2 Integrity

Integrity protection consists of both the detection of modifications to the information and the restoration of the information when a modification is detected. The integrity of cryptographic information during transit **shall** be protected using one or more of the following mechanisms:

1. Manual method (physical protection is provided):

(a) An integrity mechanism (e.g., MAC, digital signature or CRC) is used on the information, and the resulting code (e.g., MAC or digital signature) is provided to the recipient. Note: the MAC or digital signature itself may not require further integrity protection.

-OR-

(b) The information is used to perform the appropriate cryptographic operation. If the use of the keying material produces incorrect results, the received information is incorrect.

2. Electronic distribution via communication protocols (provided by the user or by the communication protocol):

(a) A cryptographic integrity mechanism (e.g., a MAC or digital signature) is used on the information, and the resulting code (e.g., MAC or digital signature) is provided to the recipient. The integrity mechanism may be applied only to the cryptographic information, or may be applied to an entire message,

-OR-

(b) The information is used to perform the appropriate cryptographic operation. If the use of the keying material produces incorrect results, the received information is incorrect.

The response to the detection of an integrity failure will vary depending on the specific environment. A security policy (see Part 2) **should** define the response to such an event. For example, if an error is detected in the received information, and the receiver requires that the information be entirely correct (e.g., the receiver cannot proceed when the information is in error), then:

a. the use of the information would be ill-advised,

b. the destruction of the information in accordance with Section 7.3.4 would be recommended,

c. the recipient may request that the information be resent (up to a predetermined number of times), and

d. information related to the incident may be stored in an audit log to later identify the source of the error.

5.2.1.3 Confidentiality

Keying material may require confidentiality protection during transit. If confidentiality protection is required, the keying material **shall** be protected using one or more of the following mechanisms:

1. Manual method:

(a) The keying material is encrypted,

-OR-

(b) The keying material is separated into key components. Each key component is handled so that no single individual can acquire access to all key components.

-OR-

(c) Appropriate physical and procedural protection is provided (e.g., by using a trusted courier).

2. Electronic distribution via communication protocols: The keying material is encrypted.

5.2.1.4 Association with Usage or Application

The association of keying material with its usage or application **shall** be either specifically identified during the distribution process or be implicitly defined by the use of the application. See Section 5.2.3 for guidance on labeling.

5.2.1.5 Association with Other Entities

The association of keying material with the appropriate entity **shall** be either specifically identified during the distribution process or be implicitly defined by the use of the application. See Section 5.2.3 for guidance on labeling.

5.2.1.6 Association with Other Related Information

Any association with other related information (e.g., domain parameters, the encryption/decryption key or IVs) **shall** be either specifically identified during the distribution process or be implicitly defined by the use of the application. See Section 5.2.3 for guidance on labeling.

5.2.2 Protection Mechanisms for Information in Storage

Cryptographic information that is not in transit is at rest in some device or storage media. This may include copies of the information that is also in transit. This information **shall** be protected in accordance with Section 5.1. A variety of protection mechanisms may be used.

The cryptographic information may be stored so as to be immediately available to an application (e.g., on a local hard disk or a server); this would be typical for keying material stored within the cryptographic module or in immediately accessible storage (e.g., on a local hard drive). The keying material may also be stored in electronic form on a removable media (e.g., a CD-ROM), in a remotely accessible location, or in hard copy form and placed in a safe; this would be typical for backup or archive storage.

5.2.2.1 Availability

Cryptographic information may need to be readily available for as long as data is protected by the information. A common method for providing this protection is to make one or more copies of the cryptographic information and store them in separate locations. During a key's cryptoperiod, keying material requiring long-term availability **should** be stored in both normal operational storage (see Section 7.2.1) and in backup storage (see Section 7.2.2.1). Cryptographic information that is retained after the end of a key's cryptoperiod **should** be placed in archive storage (see Section 7.3.1). This guideline does not preclude the use of the same storage media for both backup and archive storage.

Specifics on the long-term availability requirement for each key type are addressed for backup storage in Section 7.2.2.1, and for archive storage in Section 7.3.1.

Recovery of this cryptographic information for use in replacing cryptographic information that is lost (e.g., from normal storage), or in performing cryptographic operations after the end of a key's cryptoperiod is discussed in Sections 7.2.2.2 (recovery) and 7.3.1 (archive), and in Appendix B.

5.2.2.2 Integrity

Integrity protection is concerned with ensuring that the information is correct. Absolute protection against modification is not possible. The best that can be done is to use reasonable measures to prevent modifications, to use methods to detect (with a very high probability) any modifications that occur, and to restore the information to its original content when modifications have been detected.

All cryptographic information requires integrity protection. Integrity **shall** be provided by either physical mechanisms, cryptographic mechanisms or both.

Physical mechanisms include:

1. An Approved cryptographic module or operating system that limits access to the stored information,
2. A computer system or media that is not connected to other systems,
3. A physically secure environment that is outside a computer system (e.g., in a safe with limited access) with appropriate access controls.

Cryptographic mechanisms include:

a. A cryptographic integrity mechanism (e.g., MAC or digital signature) that is computed on the information and is later used to verify the integrity of the stored information.

b. Performing the appropriate cryptographic operation. If the use of the keying ma terial produces incorrect results, the stored information is incorrect.

If the cryptographic information needs to be restored when an error is detected, multiple copies of the information **should** be maintained in physically separate locations (i.e., in backup or archive storage; see Sections 7.2.2.1 and 7.3.1). The integrity of each copy **should** be periodically checked.

5.2.2.3 Confidentiality

One or more of the following mechanisms **shall** be used to provide confidentiality for private or secret keying material in storage: [Note that the options listed below do not provide equivalent security; probably need to recommend preferred methods or categorize.]

1. The keying material resides in an Approved cryptographic module (cryptomodule). The cryptomodule is designed to be compliant with FIPS 140-2 level 2 (or higher) and tested by an accredited laboratory of the Cryptographic Module Validation Program (CMVP). Information on this program is available at http://csrc.nist.gov/cryptval.
2. The keying material resides in an Approved cryptographic module (cryptomodule). The cryptomodule is designed to be compliant with FIPS 140-2 and has been tested by an accredited laboratory of the Cryptographic Module Validation Program (CMVP). When the cryptomodule is not is use, it is physically protected (e.g., by a locked door).
3. The keying material is encrypted and stored off-line in a secure environment (e.g., in a safe with limited access).

4. The keying material is stored off-line in a secure environment (e.g., in a safe with limited access) with access available using multi-party control procedures (i.e., two or more individuals are required for access to the keying material). It is the responsibility of each individual to assure that the confidentiality of the keying material is maintained.
5. Keying material is split into multiple key components. The key components are combined in a secure environment (e.g., in an Approved cryptomodule) when constructing the keying material. The key components are stored separately, under split knowledge procedures whereby no single individual has access to more than one key component. Each key component is protected as sensitive information (e.g., the key component is not made publicly available). One method for splitting keying material into two components, for example, is to generate a random bit string of the intended length of the components, using this bit string as one of the components, and exclusive Or-ing the bit string to the generated keying material to create the second component. However, other methods for splitting and combining components are allowed.

5.2.2.4 Association with Usage or Application

Cryptographic information is used with a given cryptographic mechanism (e.g., digital signatures or key establishment) or with a particular application. Protection **shall** be provided to ensure that the information is not used incorrectly (e.g., not only would the usage or application be associated with the keying material, but the integrity of this association would be maintained). This protection can be provided by separating the cryptographic information from that of other mechanisms or applications, or by appropriate labeling of the information. Section 5.2.3 addresses the labeling of cryptographic information.

5.2.2.5 Association with the Other Entities

Some cryptographic information needs to be correctly associated with another entity, and the integrity of this association **shall** be maintained. For example, a symmetric (secret) key used for the encryption of information, or keys used for the computation of a MAC would be associated with the other entity(ies) that shares the key. Public keys would be correctly associated (bound) with the owner of the key pair.

The cryptographic information **shall** retain its association during storage by separating the information by "entity" or application, or by properly labeling of the information. Section 5.2.3 addresses the labeling of cryptographic information.

5.2.2.6 Association with Other Related Information

An association may need to be maintained between protected information and the keying material that protected that information. In addition, keys may require association with other keying material (see Section 5.1.1).

The association is accomplished by storing the information together or providing some linkage or pointer between the information. Typically, the linkage between a key and the information it protects is accomplished by providing an identity for a key, storing the identity with the key in an identification/label, and storing the key's identity with the protected information. The association **shall** be maintained for as long as the protected information needs to be processed.

Section 5.2.3 addresses the labeling of cryptographic information.

5.2.3 Labeling of Cryptographic Information

Labels may be used with cryptographic information to define the use of that information or to provide a linkage between cryptographic information.

5.2.3.1 Labels for Keys

A label may be used to provide information for the use of a key. Different applications may require different labels for the same key type, and different labels may be required for different key types. It is the responsibility of an implementer to select a suitable label for a key. When labels are used, the label **should** accompany a key (i.e., is typically stored or transmitted with a key) and contain some subset of the following information:

1. Identity of the key
2. Association with other keying material (e.g., it is recommended that a public and private key be associated with each other)
3. Identity of the key's owner or the sharing entity
4. Cryptoperiod (e.g., start date and end date)
5. Key type (e.g., signing private key, encryption key, master key)
6. Application (e.g., purchasing, email)
7. Counter
8. Domain parameters (e.g., the domain parameters used by DSA or ECDSA, or a pointer to them)
9. Modulus
10. Key encrypting key (e.g., identity of key wrapping key, algorithm of key wrapping key, etc.)
11. Integrity protection mechanism (e.g., key and algorithm used to provide cryptographic protection, and protection code (e.g., MAC, digital signature))
12. Other information (e.g., length of the key, protection requirements, who has access rights, additional conditions for use)

5.2.3.2 Labels for Related Cryptographic Information

Cryptographic information other than keying material may need a label to "point to" the keying material that was used to provide the cryptographic protection for the information. The label may also contain other related cryptographic information. When labels are used, the label **should** accompany the information (i.e., is typically stored or transmitted with the information) and contain some subset of the following information:

1. The type of information (e.g., domain parameters)
2. Source of the information (e.g., the entity that sent the information)
3. Application (e.g., purchasing, email)
4. Other associated cryptographic information (e.g., a key, MAC or hash value)
5. Any other information (e.g., who has access rights)

6. Key States

The life cycle of a cryptographic key consists of several states between its generation and its destruction. These states are defined from a system point of view, as opposed to a single cryptomodule point of view. In some cases, a key may exist in multiple states.

1. **Pre-activation state**: The key has been generated, but is not yet active. This is the state of a key during its distribution, though the state may extend beyond distribution. A key **shall not** be used to apply cryptographic protection to information (e.g., encrypt or sign information to be transmitted or stored) while in this state. The key also **shall not** be used to process cryptographically protected information (e.g., decrypt ciphertext or verify a digital signature) while in this state.
2. **Active state**: The key may be used to apply cryptographic protection to information or to process cryptographically protected information (e.g., decrypt ciphertext or verify a digital signature) or both.
3. **Process only state**: The key may be used to process cryptographically protected information, but **shall not** be used to apply cryptographic protection to information.
4. **Deactivated state**: The key **shall not** be used to either apply cryptographic protection to information, or to process cryptographically protected information. The key is in normal operational or backup storage. Some keys may be archived from this state.
5. **Archive state**: The key is archived and **shall only** be used to process cryptographically protected information. The security of the key and its use in this state is dependent on the security provided by the archive. A key in the archive state is also considered to be in either the deactivated or compromised state. If the key is also in the compromised state, it may not be appropriate to use the key unless it was archived prior to its compromise and a decision is made to "honor" the cryptographic processing (e.g., because the compromise occurred after the digital signature on signed information was applied). When a key in the archive state is no longer required for processing cryptographically protected information, the key **should** be destroyed (see Section 7.3.4).
6. **Compromised state**: The integrity or secrecy of the key is suspect. This state may be entered from any other state. A revocation, for example, would cause a key to enter this state. In certain cases, a compromised key may be used to process cryptographically protected information.
7. **Destroyed state**: The key **shall** be destroyed as specified in Section 7.3.4.

Transitions between states are triggered by events, such as a date or the compromise of a key. Figure 1 depicts the key states and the corresponding transition events.

Transition 1:	A key enters the pre-activation state immediately upon generation.
Transition 2:	Assuming that the key is not deactivated or determined to be compromised while in the pre-activation state (see transitions 5 and 8), a key transitions from the pre-activation state to the active state in accordance with an *activation date* or, if no activation date is available, as soon as the key can be used (e.g., as soon as it is available).

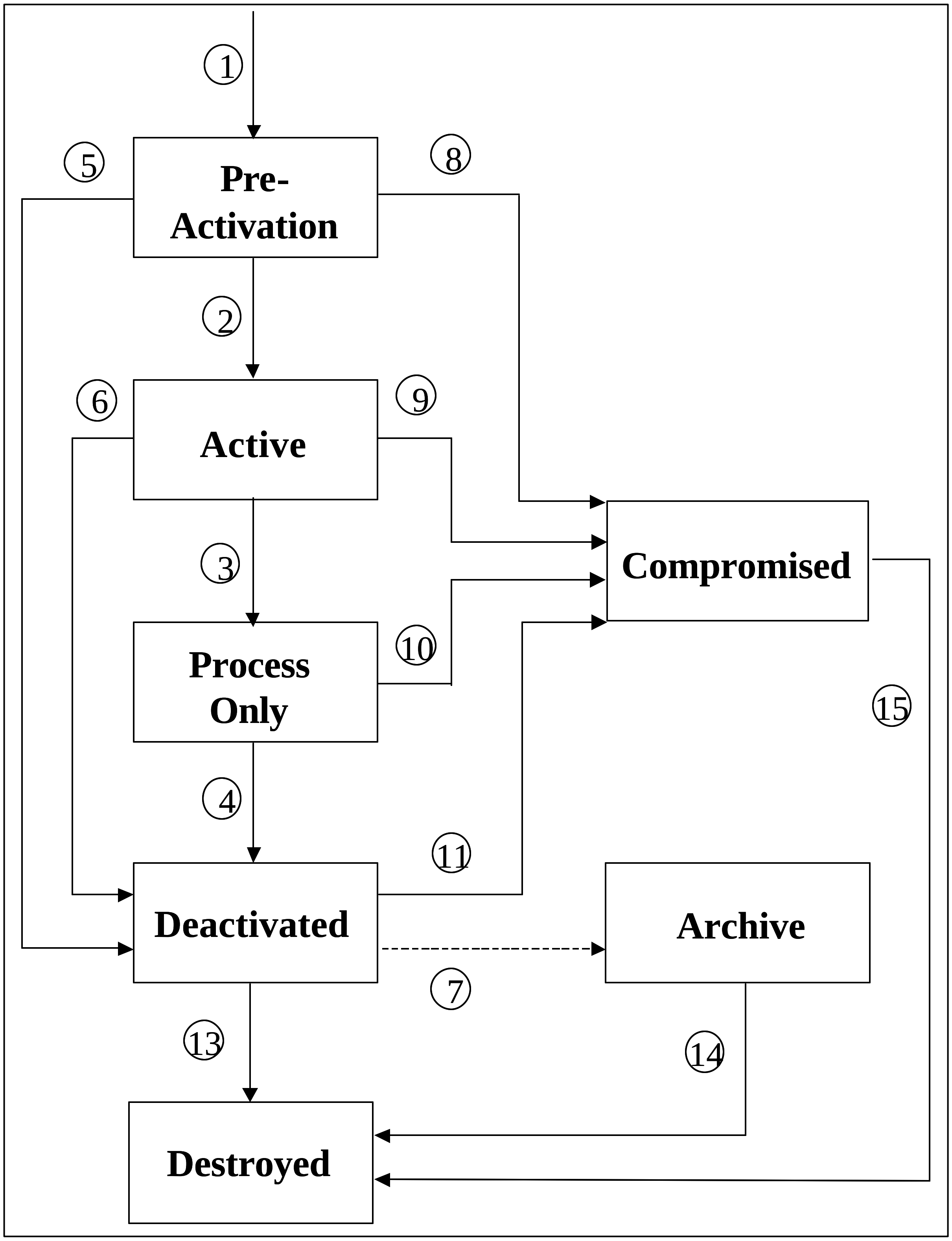

Figure 1: Key States

For example, in many cases an activation date (e.g., the validity period begin date) may not be associated with a digital signature key pair. However, it is recommended that the key pair not be considered as active until a certificate has been issued. When the certificate is issued, the key pair is then active.

Note that in some cases the period of time during which a key is in the pre-activation state may be very short. For example, during a key agreement process, the agreed upon key may be considered as immediately active.

The beginning of a key's cryptoperiod is considered to be the time that the key enters the active state (see Section 8.2).

Transition 3: Assuming that a key is not deactivated or determined to be compromised while in the active state (see transitions 6 and 9), a key may transition to the process only state in accordance with a *process only date* (either explicitly or implicitly specified).

For example, a symmetric data encryption key may be used for a predetermined period of time to both encrypt and decrypt information, but when the process only date is reached, the key is not be used for further encryption, but may continue to be used for decryption.

For some keys and applications, there is no need for a process only state, or the active state and the process only state can be considered to be coincident. In this case, transition 3 is moot. This may be the case, for example, for a data encryption key used for the protection of transmitted information during a single communication session.

Transition 4: Assuming that a key is not determined to be compromised while in the process only state, a key may transition from the process only state to the deactivated state in accordance with a *deactivation date*. This is often referred to as the end of a key's cryptoperiod (see Section 8.2).

Transitions 5 and 6: A key may transition from the pre-activation state or the active state to the deactivated state as a result of a revocation action (see Section 7.3.5) for a reason other than a key compromise, or if the key is replaced (see Section 7.2.3).

Transition 7: A key may transition from the deactivated state (or any state other than the compromised state) to the archive state when the key may be required for processing cryptographically protected information after the key has been deactivated, i.e., after the end of the key's cryptoperiod. Not all keys make this transition (see Sections 7.3.1 and Appendix B.3).

Transitions 8-12: A key **shall** transition to the compromised state when the integrity or confidentiality of a key requiring confidentiality protection is suspect.

Transitions 13-15: A key **should** transition to the destroyed state as soon as no longer needed.

Symmetric keys may pass through the pre-activation, active, process only and deactivated states, and possibly the archive state. If compromised, a symmetric key would enter the compromised state.

Asymmetric keys that are or will be certified are in the pre-activation state until certified or until the "not before" date has passed. The types of transitions for asymmetric keys depend on the key type. Examples of transitions follow:

a. A private signature key transitions from the active state to the deactivated state. This key does not enter the process only state, since it is never used to process "received information". A private signature key may transition to the compromised state, but it is not recommended that the key transition to the archive state (see Section 7.3.1).

b. A public signature verification key could be thought of as having a coincident active and process only state. Once deactivated (e.g., by the end of the validity period, or upon revocation), it enters the deactivated state and could enter the archive state. The public signature verification key enters the compromised state only if its integrity becomes suspect.

c. A private key transport key could be thought of as having a coincident active and process only state. Once deactivated (e.g., by the end of the validity period), it enters the deactivated state. It could also enter both the archive state and the compromised state.

d. A public key transport key could be thought of as having a coincident active and process only state. Once deactivated (e.g., by the end of the validity period, or upon revocation), it enters the deactivated state. It does not enter the archive state (see Section 7.3.1), and enters the compromised state only when its integrity is suspect.

e. Private and public key agreement keys have coincident active and process only states. They may enter the deactivated state, but do not normally enter the archive state (see Section 7.3.1). The keys may enter the compromised state, but the public key would only do so if its integrity were suspect.

7 Key Management Phases

Cryptographic key management encompasses the entire life cycle of cryptographic keys and other keying material from a system or enterprise perspective. In addition to the key states defined in Section 6, cryptographic key management includes the establishment and management of any associated attributes.

These attributes are leveraged by applications to select the appropriate cryptographic key(s) for a particular service. Attributes required for key management might include the identity of a person or system associated with that key or the types of information that person is authorized to access. While these attributes do not appear in cryptographic algorithms, they are crucial to the implementation of applications and application protocols.

The following subsections discuss the various functions of key management. The functions may be divided into various phases that relate to the key states discussed in Section 6.

1. Pre-operational phase: The keying material is not yet available for normal cryptographic operations. Keys may not yet be generated, or may be in the pre-activation state. System or enterprise attributes are established during this phase as well.
2. Operational phase: The keying material is available and in normal use. Keys are in the active or process only states.
3. Post-operational phase: The keying material is no longer in normal use, but access to the keying material is possible. Keys are in the archived state; the keys may also be in the compromised state.
4. Obsolete/destroyed phase: The keying material is no longer available. All records of its existence may have been deleted. Keys are in the de-activated, compromised, or destroyed state.

A key management system may not have all identified functions, since some functions may not be appropriate. In some cases, one or more functions may be combined, or the functions may be performed in a different order. For example, a system may omit the post-operational phase if keys are never archived. In this case, keys would move from the operational phase directly to the obsolete/destroyed phase.

When functions are combined, the keying material **should** be handled as though the functions are separate (e.g., if keying material is only to be retained during the key's cryptoperiod but no longer, it is not recommended that the keying material be removed from normal operational and backup storage at the end of the cryptoperiod and not retained for archive purposes).

A flow diagram for the key management phases is presented as Figure 2. Five phase transitions are identified in the diagram:

1. After the required attributes have been established, keying material has been generated, and the attributes are associated with the key during the pre-operational phase, the key is ready to use in applications and transitions to the operational phase at the appropriate time.

2. When keys are no longer required for normal use (i.e., the end of the cryptoperiod has been reached and the key is no longer "active"), but access to those keys needs to be maintained, the key transitions to the post-operational phase.
3. When an application processes "old" data, it may require the use of a currently inactive key (i.e., a key whose cryptoperiod has passed). This key may transition from the post-operational phase back into the operational phase. Typically, the key will transition back to the post-operational phase immediately after processing the old data.
4. Some applications will require that access be preserved for a period of time, then the keying material may be destroyed. When it is clear that a key in the post-operational phase is no longer needed, it may transition to the obsolete/destroyed phase.
5. Access does need not be preserved for some keys for any length of time after use. In this case, the keys may transition directly from the operational phase to the obsolete/destroyed phase.

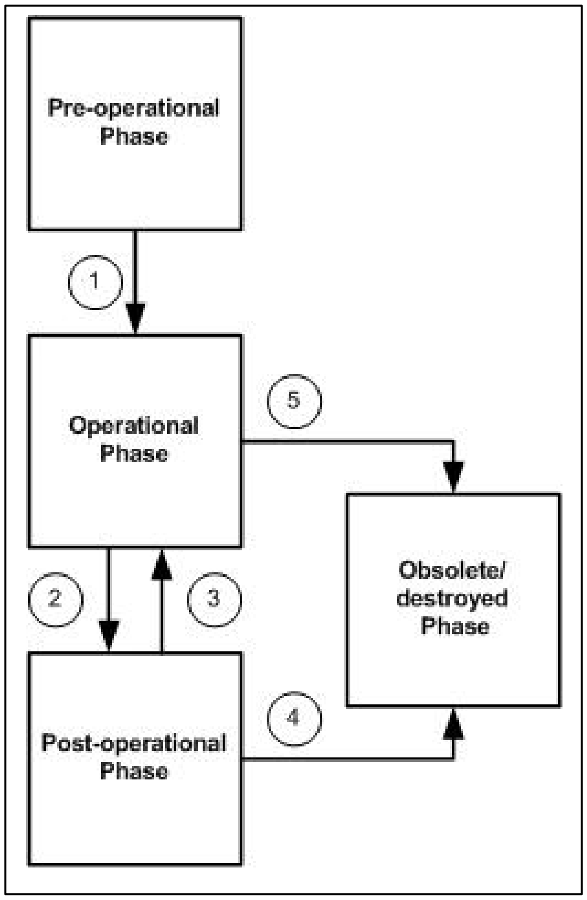

Figure 2: Key Management Phases

7.1 Pre-operational Phase

During the pre-operational phase of key management, keying material is not yet available for normal cryptographic operations.

7.1.1 User Registration

During user registration, an entity interacts with a registration authority to become an authorized member of a security domain. In this phase, a user or device name may be established to identify the member during future transactions. In particular, security infrastructures may associate the identification information with the entity's keys (see Section 7.1.6, Key Registration). The entity may also establish various attributes during the registration process, such as email addresses or role/authorization information. As with identity information, these attributes may be associated with the entity's keys by the infrastructure to support secure application-level security services.

Since applications will depend upon the identity established during this process, it is crucial that the registration authority establish appropriate procedures for the validation of identity. Identity may be established through an in-person appearance at a registration authority, or may be established entirely out-of-band. The strength (or weakness) of a security infrastructure will often depend upon the identification process.

User and key registration (see Section 7.1.6) may be performed separately, or in concert. If performed separately, the user registration process will generally establish a secret value (e.g., a password, PIN, or HMAC key); the secret value may be used to authenticate the user during the key registration step. If performed in concert, the user establishes its identity and performs key registration in the same process. In this case, a secret value is unnecessary.

7.1.2 System Initialization

System initialization involves setting up or configuring a system for secure operation. This may include algorithm preferences, the identification of trusted parties, and the definition of domain parameter policies and any trusted parameters (e.g., recognizing certificate policies or the identification of IP address validation servers).

7.1.3 User Initialization

User initialization consists of an entity initializing its cryptographic application (e.g., installing and initializing software or hardware). This involves the use or installation (see Section 7.1.4) of the initial keying material that may be obtained during user registration. Examples include the installation of a key at a CA, trust parameters, policies, trusted parties, and algorithm preferences.

7.1.4 Keying Material Installation

The security of keying material installation is crucial to the security of a system. For this function, keying material is installed for operational use within an entity's software, hardware, system, application, cryptomodule, or device using a variety of techniques. Keying material is installed when the software, hardware, system, application, cryptomodule, or device is initially set up, when new keying material is added to the existing keying material, and when existing keying material is replaced (via rekeying, key update, or key derivation - see Section 7.2.3).

The process for the initial installation of keying material (e.g., by manual entry, electronic key loader, by a vendor during manufacture) **shall** include the protection of the keying material during entry into a software/hardware/system/application/cryptomodule/ device, take into account the requirements of FIPS 140-2 and its differing requirements based on levels of protection, and include any additional procedures that may be required.Note: Should this section contain guidance on:

- issues related to the installation of additional keying material,
- issues related to replacing existing keying material that are not covered in Section 7.2.3,
- issues related to keying material installation during key recovery that are not addressed in Sections 7.2.2.2 or 7.3.1, or in Appendix B.

Many applications or systems are provided by the manufacturer with keying material that is used to test that the newly installed application/system is functioning properly. This test keying material **shall** not be used operationally.

7.1.5 Key Establishment

Key establishment includes the generation of keying material by an entity for its own use (e.g., for the protection of stored information); and the generation and distribution, or the agreement of keying material for communication between entities. During this process, some of the keying material may be in transit (e.g., the keying material is being manually or electronically distributed). Other keying material may be retained locally. In either case, the keying material **shall** be protected in accordance with Section 5.1.

An entity may be an individual (person), organization, device or process. When keying material is generated by an entity for its own use, and the keying material is not distributed among "sub-entities" (e.g., is not distributed among various individuals, devices or processes within an organization), one or more of the appropriate protection mechanisms for stored information in Section 5.2.2 **shall** be used.

Keying material that is distributed between entities or between sub-entities of a single entity **shall** be protected using one or more of the appropriate protection mechanisms specified in Section 5.2.1. Any keying material that is not distributed (e.g., the private key of a key pair, or one's own copy of a symmetric key) **shall** be protected using one or more of the appropriate protection mechanisms specified in Section 5.2.2.

7.1.5.1 Generation and Distribution of Asymmetric Key Pairs

Key pairs **shall** be generated in accordance with the mathematical specifications of the appropriate Approved standard.

A static key pair **shall** be generated within a FIPS 140-2 validated cryptographic module or in a facility that is approved for the generation of classified keying material.. A static key pair **shall** be either generated by the entity that "owns" the key pair (i.e., the entity that uses the private key in the cryptographic computations) or by a facility that distributes the key pair in accordance with Section 7.1.5.1.3. When generated by the entity that owns the key pair, the signing private key **shall not** be distributed to other entities. In the case of a signature verification public key and its associated private key, the owner **should** generate the keying material rather than any other entity generating the keying material for that owner; this will facilitate non-repudiation.

Ephemeral keys are often used for key establishment (see [SP 800-56]). They are short-lived and are statistically unique to each execution of a key establishment process (e.g., unique to each message or session). An ephemeral key pair **shall** be generated within a FIPS 140-2 validated cryptographic module.

The generated key pairs **shall** be protected in accordance with Section 5.1.1.

7.1.5.1.1 Distribution of Static Public Keys

Static public keys are relatively long lived and are typically used for a number of executions of an algorithm. The distribution of the public key **should** provide assurance to the receiver of that key that:

1. the true owner of the key is known (i.e., the owner of the key pair); this requirement may be disregarded if anonymity is acceptable. However, the strength of the cryptographic system and trust in the validity of the protected data depends, in large part, on the assurance of the public key owner' identity,

2. the purpose/usage of the key is known (e.g., RSA digital signatures or elliptic curve key agreement),
3. any parameters associated with the public key are known (e.g., domain parameters), and
4. the public key has been properly generated (e.g., the owner of the public key actually has the associated private key and assurance of public key va lidity has been obtained).

Keys falling into this category are:

a. The *public signature verification key,*
b. The *public authentication key,*
c. The *public key transport key,*
d. The *public static key agreement key,* and
e. The *public authorization key*.

7.1.5.1.1.1 Distribution of a Trust Anchor's Public Key in a PKI

The public key of a trusted CA, or trust anchor, is the foundation for all PKI-based security services. The trust anchor is not a secret, but the *authenticity* of the trust anchor is the crucial assumption for PKI. Trust anchors may be obtained through many different mechanisms, providing different levels of assurance. The types of mechanisms that are provided may depend on the role of the user in the infrastructure. A user that is only a “relying party” – that is, a user that does not have keys registered with the infrastructure – may use different mechanisms than a user that possesses keys registered by the infrastructure. Trust anchors are frequently distributed as "self-signed" X.509 certificates, that is, certificates that are signed by the subject public key of the certificate.

Trust anchors are often embedded within an application and distributed with the application. For example, the installation of a new web browser typically includes the installation or update for the user’s trust anchor list. Operating systems often are shipped with "code signing" trust anchor public keys. The user relies upon the authenticity of the software distribution mechanism to ensure that only valid trust anchors are installed during installation or update. However, in some cases other applications may install trust anchor keys in web browsers.

Trust anchors in web browsers are used for several purposes, including the validation of S/MIME e-mail certificates and web server certificates for "secure websites" that use the SSL/TLS protocol to authenticate the web server and provide confidentiality. Relying party users who visit "secure" websites that have a certificate not issued by a trust anchor CA may be given an opportunity to accept that certificate, either for a single session, or permanently. It is also recommended that relying parties be cautious about accepting certificates from unknown certification authorities so that they do not, in effect, inadvertently add new permanent trust anchors.

Roaming users may have particular concerns about trust anchors used by web browsers when they use systems in kiosks, libraries, Internet cafes, or hotels and systems provided by conference organizers to access "secure websites." The user than has no control over the trust anchors installed in the host system, and simply trusts the host systems to make sound choices of trust anchors. Trust anchor distribution through software installation does not require that the relying party be a direct participant in the infrastructure. The relying party trusts the software distribution mechanism to avoid installation of malicious code. Extending this trust to cover trust anchors for that application may be reasonable, and allows the relying party to obtain trust anchors without any additional procedures.

Where a user is a direct participant in the infrastructure, additional mechanisms are usually available. The user interacts with the infrastructure to register its keys (e.g., to obtain certificates), and these interactions may be extended to provide trust anchor information. This allows the user to establish trust anchor information with approximately the same assurance that the infrastructure has in the user's keys. In the case of a PKI:

1. The initial distribution of the public key of a trusted CA, or trust anchor, **should** be performed in conjunction with the presentation of a requesting entity's public key to a registration authority (RA) or CA during the certificate request process. In general, the CA's public key, associated parameters, key use, and proof-of-possession are conveyed as a self-signed X.509 public key certificate. The certificate is digitally signed by the private key that corresponds to the public key within the certificate. While parameters, and proof-of-possession may be clearly established from the self-signed certificate, the CA's identity information cannot be verified from the certificate itself.
2. The trusted process used to convey a requesting entity's public key and assurances to the RA or CA **shall** also protect the trust anchor information conveyed to the requesting entity. In cases where the requesting entity appears in person, the trust anchor information may be provided at that time. If a secret value has been established during user registration (see Section 7.1.1), the trust anchor information may be supplied with the requesting entity's certificate. In cases where the RA or CA delegates the verification of the requesting entity's identity to another trusted process, the trust anchor information **should** be provided along with the unique, unpredictable information (see Section 7.1.1.2).

7.1.5.1.1.2 *Submission to a Registration Authority or Certification Authority*

Public keys may be provided to a Certification Authority (CA) or to a registration authority (RA) for subsequent certification by a CA. During this process, the RA or CA **should** be provided with the assurances listed in Section 7.1.5.1.1 by the owner of the key or an authorized representative (e.g., the firewall administrator). The assurances include the owner's identity, the key use, any parameters to be used, and assurance of public key validity.

In general, the owner of the key is identified in terms of an identity established during user registration (see Section 7.1.1). The key owner identifies the appropriate uses for the key, along with the parameters and any assurances of validity. In cases where anonymous ownership of the public key is acceptable, the owner or the registration authority generates a pseudonym to be used as the identity.

In addition to the public key and the assurances listed above, proof of possession (POP) of the corresponding private key **shall** be provided by the reputed owner of the key pair. If the owner is not required to perform POP, it is possible that the CA would bind the public key to the wrong entity. As a general rule, the owner **should** prove possession by performing operations with the private key that satisfy the indicated key use. For example, if a key pair is intended to support key transport, it is recommended that the owner decrypt a key provided to the owner by the CA that is encrypted using the owner's public key. If the owner can correctly decrypt the ciphertext key using the associated private key, then the owner has established POP. In this case, it is not recommended that POP be afforded by generating and verifying a digital signature with the key pair.

As with user registration, the strength of the security infrastructure depends upon the methods used for distributing the key to a RA or CA. There are many different methods, each appropriate for some range of applications. Some examples of common methods are:

1. The public key, assurances and POP are provided by the public key owner in person, or by an authorized representative of the private key owner.

2. The identity of the public key owner is established at the RA or CA in person or by an authorized representative of the public key owner during user registration. Unique, unpredictable information (e.g., an authenticator or cryptographic key) is provided at this time by the RA or CA to the owner as a secret value. The public key, remaining assurances (key use, parameters, and assurance of public key validity), and POP are provided to the RA or CA using a communication protocol protected by the secret value. It is recommended that the secret value be destroyed by the key owner as specified in Section 7.3.4 upon receiving confirmation that the certificate has been successfully generated. The RA or CA may maintain this information for auditing purposes, but it is recommended that the RA or CA not accept further use of the unique identifier to prove identity.

   When a specific list of public key owners are pre-authorized to register keys, identifiers may be generated in bulk. In this case, it is critical to protect the secret value from disclosure, and the procedures must demonstrate that the chain of custody was maintained. The secret value’s lifetime should be limited, but must allow for the public key owner to appear at the RA or CA, generate their keys, and provide the public key (under the secret value’s protection) to the RA or CA. Since it may take some time for the public key owner to appear at the RA or CA, a two or three week lifetime is probably reasonable.

   When public key owners are not pre-authorized, the RA or CA must generate the identifier in the user’s presence. In this case, the time limit may be much more restrictive, since the public key owner need only generate their keys and provide the public key to the CA or RA. In this case, a 24-hour lifetime for the secret value would be reasonable.

3. The identity of the public key owner is established at the RA or CA using a previous determination of the public key owner’s identity. This is accomplished by “chaining” a new public key certificate request to a previously certified pair (i.e., the signature verification public key). For example, the request for a new public key certificate is signed by the owner of the new public key to be certified. It is recommended that the signing private key used to sign the request be associated with a verification public key that is certified by the same CA that will certify the new public key. It is also recommended that the request contain the new public key, remaining assurances (key use, parameters, and validation information), and POP.

4. The public key, key use, parameters, validity assurance information, and POP are provided to the RA or CA along with a claimed identity. The RA or CA delegates the verification of the public key owner's identity to another trusted process (e.g., an examination of the public key owner's identity by the U.S. postal service when delivering registered mail). Upon receiving a request for certification, the RA or CA generates and sends unique, unpredictable information (e.g., an authenticator or cryptographic key) to the requestor using the trusted process (e.g., a courier). The trusted process verifies the identity of the requestor prior to delivery of the information provided by the RA or CA. The owner uses this information to prove that the trusted process succeeded, and the RA or CA delivers the certificate to the owner. It is recommended that the information be destroyed by the key owner as specified in Section 7.3.4 upon receiving confirmation that the certificate has been successfully generated. (The RA or CA may maintain this information for auditing purposes, but should not accept further use of the unique identifier to prove identity.)

In cases involving an RA, upon receipt of all information from the requesting entity (i.e., the owner of the new public key), the RA forwards the relevant information to a CA for certification. The CA **shall** perform any validation or other checks required for the algorithm with which the public key will be used (e.g., public key validation) prior to issuing a certificate. The CA **should** indicate the checks or validations that have been performed (e.g., in the certificate, or in the CA policy or practices statement). After generation, the certificate is distributed manually or electronically to the RA, the public key owner, or a certificate repository (i.e., a directory) in accordance with the CA's certificate practices statement.

7.1.5.1.1.3 *General Distribution*

Public keys may be distributed to entities other than an RA or CA in several ways. Distribution methods include:

1. Manual distribution of the public key itself by the owner of the public key (e.g., in a face to face transfer, or by a bonded courier); the assurances listed in Section 7.1.5.1.1 **should** be provided to the recipient prior to the use of the public key.
2. Manual (e.g., in a face to face transfer or by receipted mail) or electronic distribution of a public key certificate by the public key owner, the CA, or a certificate repository (i.e., a directory). Assurances listed in Section 7.1.5.1.1 that are not provided by the CA (e.g., public key validation) **should** be provided to or performed by the receiver of the public key prior to the use of the key.
3. Electronic distribution of a public key (e.g., using a communication protocol with authentication and content integrity) in which the distributed public key is protected by a certified key pair owned by the entity distributing the public key. The assurances listed in Section 7.1.5.1.1 **should** be provided to the receiving entity prior to use of the public key.

7.1.5.1.2 Distribution of Ephemeral Public Keys

When used, ephemeral public keys are distributed as part of a secure key agreement protocol. The key agreement process (i.e., the key agreement scheme + the protocol + any associated negotiation) **should** provide a recipient with the assurances listed in Section 7.1.5.1.1. The recipient of an ephemeral public key **shall** immediately obtain assurance of validity of that key as specified in [SP 800-56] prior to continuing with a key agreement process.

7.1.5.1.3 Distribution of Centrally Generated Key Pairs

When a static key pair is centrally generated, the key pair **shall** be generated within a FIPS 140-2 validated cryptographic module or in a facility that is approved for the generation of classified keying material as specified in Section 7.1.5.1 for subsequent delivery to the intended owner of the key pair. A signing key pair generated by a central key generation facility for its subscribers will not provide strong non-repudiation for those individual subscribers; therefore, when non-repudiation is required by those subscribers, the subscribers **should** generate their own signing key pairs. However, if the central key generation facility generates signing key pairs for its own organization and distributes them to members of the organization, then non-repudiation is provided at an organizational level (but not an individual level).

The private key of a key pair generated at a central facility **shall** only be distributed to the intended owner of the key pair. The confidentiality of the centrally generated private key **shall** be protected, and the procedures for distribution **shall** authenticate the recipient's identity as established during user registration (see Section 7.1.1).

The key pair may be distributed to the intended owner using an approved manual method (e.g., courier, mail or other method specified by the key generation facility) or secure electronic method (e.g., a secure communication protocol). The private key **shall** be distributed in the same manner as a symmetric key (see Section 7.1.5.2.2). During the distribution process, each key of the key pair **shall** be provided with the appropriate protections for that key (see Section 5.1).

When split knowledge procedures are used for the manual distribution of the private key, the key **shall** be split into multiple key components that are the same length as the original key; each key component **shall** provide no knowledge of the original key (e.g., each key component **shall** appear to be generated randomly).

Upon receipt of the key pair, the owner **should** validate that the public and private keys of the key pair are correctly associated (i.e., check that they work together, for example, checking that the signing private key can be verified by the verification public key), and obtain assurance of the validity of the public key (see [SP 800-56] and [FIPS 186-3]. Further certification and/or distribution of the public key **should not** be performed until these checks have been made.

7.1.5.2 Generation and Distribution of Symmetric Keys

The symmetric keys used for the encryption and decryption of data or other keys and for the computation of MACs (see Sections 4.2.2 and 4.2.3) **shall** be determined by an Approved method and **shall** be provided with protection that is consistent with Section 5.

Symmetric keys **shall** be:

1. generated and subsequently distributed (see Sections 7.1.5.2.1 and 7.1.5.2.2) either manually (see Section 7.1.5.2.2.1), using a public key transport mechanism (se Section 7.1.5.2.2.2), or using a previously distributed or agreed upon key encrypting key (see Section 7.1.5.2.2.2),
2. established using a key agreement scheme (i.e., the generation and distribution are accomplished with one process) (see Section 7.1.5.2.3),
3. determined by a key update process (see Section 7.2.3.2), or
4. derived from a master key (see Section 7.2.4).

7.1.5.2.1 Key Generation

Symmetric keys determined by key generation methods **shall** be either generated by an Approved random number generation method, created from the previous key during a key update procedure (see Section 7.2.3.2), or derived from a master key (see Section 7.2.4). Note that in some applications, keys are generated from passwords; the use of these keys to protect information has not been evaluated. See Appendix C for a discussion on the generation of keys from passwords..

When split knowledge procedures are used for the manual distribution of the key (see Section 7.1.5.2.2.1), the key **shall** exist as multiple key components that are the same length as the original key; each key component **shall** provide no knowledge of the key (e.g., each key component must appear to be generated randomly and have no identifiable relationship to the key). The keying material may be created and then split into components, or may be created as separate components.

Key generation **shall** be performed in a FIPS 140-2 validated cryptographic module or in a facility that is approved for the generation of classified keying material.

Keys used only for the storage of information (i.e., data or keying material) **should not** be distributed except for backup or to other entities within an organization that may require access to the information protected by the keys. If the keys are used to protect data or keys to be communicated between entities, one of the entities generates the keys and transports (sends) the keys to the other entity(ies) (see Section 7.1.5.2.2).

7.1.5.2.2 Key Distribution

Keys generated in accordance with Section 7.1.5.2.1 as key encrypting keys (used for key wrapping), as the initial key for key update, as master keys to be used for key derivation, or for the protection of communicated information are distributed manually (manual key transport) or using an electronic key transport protocol (electronic key transport).

7.1.5.2.2.1 Manual Key Distribution

Keys distributed manually (i.e., by other than an electronic key transport protocol) **shall** be protected throughout the distribution process. During manual distribution, secret or private keys **shall** either be encrypted or be distributed using appropriate physical security procedures. If multi-party control is desired, split knowledge procedures may be used as well. The manual distribution process **shall** assure:

1. that the distribution of the keys is authorized,
2. that the entity distributing the keys is trusted by both the entity that generates the keys and the entity(ies) that receives the keys,
3. the keys are protected in accordance with Section 5, and
4. that the keys are received by the authorized recipient.

When distributed in encrypted form, the key **shall** be encrypted by a key encrypting key that is used only for key wrapping, or by a public key transport key owned by the intended recipient. The key encrypting key or public key transport key **shall** have been distributed as specified in this guideline.

When using split knowledge procedures, each key component **shall** be either encrypted or distributed separately via secure channels for delivery to a different individual. Appropriate physical security procedures **shall** be used to protect each key component as sensitive information.

Physical security procedures may be used for all forms of manual key distribution. However, these procedures are particularly critical when the keys are distributed in plaintext form. In addition to the assurances listed above, accountability and auditing of the distribution process (see Sections 8.5 and 8.6) **should** be used.

7.1.5.2.2.2 Electronic Key Distribution/Key Transport

Electronic key distribution, or key transport, is used to distribute keys via a communication channel (e.g., the Internet or a satellite transmission). Electronic key transport requires the prior distribution of a key encrypting key to be used for key wrapping or a public key transport key as follows:

1. A key encrypting key used for key wrapping **shall** be generated and distributed in accordance with Sections 7.1.5.2.1 and 7.1.5.2.2, or established using a key agreement scheme as defined in Section 7.1.5.2.3.
2. A public key transport key **shall** be generated and distributed as specified in Section 7.1.5.1.

Approved schemes for the transport of keys using the previously established key encrypting key or public key transport key are provided in [SP 800-56]. The Approved key transport schemes provide assurance that:

a. the key encrypting key and the distributed key are not disclosed or modified, and

b. the keys are protected in accordance with Section 5.

In addition, the Approved key transport schemes, together with the associated key establishment protocol, **should** provide assurance to the recipient that the recipient has received the correct key. In some protocols, the assurance is extended to the sender as well.

7.1.5.2.3 Key Agreement

Key agreement is used in a communication environment to establish keying material using public information contributed by all entities in the communication (most commonly, only two entities) without actually sending the keying material. Approved key agreement schemes are provided in [SP 800-56]. Key agreement requires the availability of asymmetric key pairs or symmetric key encrypting keys (i.e., key wrapping keys) that are used to calculate shared secrets, which are then used to derive symmetric keys and other keying material (e.g., IVs).

A key agreement scheme uses (1) symmetric key encrypting keys, or (2) either static or ephemeral key pairs or both. The key pairs **should** be generated and distributed as discussed in Section 7.1.5.1. The static and ephemeral key pairs and the subsequently derived keying material must be protected as specified in Section 5.

A key agreement scheme and its associated key establishment protocol **should** provide the following assurances:

1. Each entity in the key establishment process knows the identity of the other entity(ies); this may be achieved by the key agreement scheme or may be achieved by the protocol in which key agreement is performed[8]. Note that the identity may be a "pseudo-identity", not the identity appearing on the entity's birth certificate, for example.
2. The keys used in the key agreement scheme are correctly associated with the entities involved in the key establishment process.
3. The derived keys are correct.

Keys derived through key agreement and its enabling protocol **should not** be used to protect and send information until all desired properties have been achieved.

7.1.5.3 Generation and Distribution of Other Keying Material

Keys are often generated in conjunction with or are used with other keying material. This other keying material **shall** be protected in accordance with Section 5.2.

7.1.5.3.1 Domain Parameters

Domain parameters are used by some public key algorithms to generate key pairs, to compute digital signatures, or to establish keys. Typically, domain parameters are generated infrequently and used by a community of users for a long period of time. Domain parameters may be distributed in the same manner as the public keys with which they are associated, or they may be made available at some other accessible site. Assurance of the validity of the domain parameters **shall** be obtained prior to use, either by a trusted entity that indicates its "blessing" on the parameters (e.g., a CA), or by the entity's themselves. Assurance of domain parameter validity is addressed in [FIPS 186-3] and [SP 800-56]. Obtaining this assurance **should** be addressed in a CA's certificate practices statement or an organization's security plan.

7.1.5.3.2 Initialization Vectors

Initialization vectors (IVs) are used by symmetric algorithms in several modes of operation for encryption and decryption, or for authentication. The criteria for the generation and use of IVs is provided in [SP 800-38A]; IVs **shall** be protected as specified in Section 5. When distributed, IVs may be distributed in the same manner as their associated keys, or may be distributed with the information that uses the IVs as part of the encryption or authentication mechanism.

7.1.5.3.3 Shared Secrets

Shared secrets are computed during a key agreement process and are subsequently used to derive keying material. Shared secrets are generated as specified by the appropriate key agreement scheme (see [SP 800-56]), but **shall not** be distributed.

[8] The identity of another entity is established (a) by a key agreement scheme when that entity uses a static key pair that has been registered with an infrastructure (e.g., a PKI), or (b) by the use of a key that has been previously established with the receiving entity. In case (b), the previously established key could be used to authenticate that the entity is the same as the entity associated with previous transactions, without ever revealing the entity's real identity. This may be beneficial in special circumstances, such as a whistleblower scenario.

7.1.5.3.4 Secret and Public Seeds

Seeds are used to initialize a pseudorandom number generator (PRNG). The criteria for the selection of a seed is provided in the specification of an Approved PRNG. Secret seeds **shall not** be distributed; public seeds may be distributed using a method that will protect the public seed as specified in Section 5.

7.1.5.3.5 Intermediate Results

Intermediate results occur during computation using cryptographic algorithms. These results **should not** be distributed.

7.1.6 Key Registration

Key registration results in the binding of keying material to information or attributes associated with a particular entity. This information typically includes the identity of the entity associated with the keying material and the intended use of the keying material (e.g., signing key, data encryption key, etc.). Additional information may include authorization information or specify the level of trust. The binding provides assurance to the community at large that the keying material is used by the correct entity in the correct application. The binding is often cryptographic, which creates a strong association between the keying material and the entity. A trusted third party performs the binding. Examples of a trusted third party include a Kerberos realm server or a PKI certification authority (CA).

When a Kerberos realm server performs the binding, a symmetric key is stored on the server with the corresponding attributes, In this case, the registered keying material is maintained in confidential storage (i.e., the keys are provided with confidentiality protection).

When a CA performs the binding, the public key and associated attributes are placed in a public key certificate, which is digitally signed by the CA. In this case, the registered key material may be publicly available.

When a CA provides a certificate for public keys, the keys **shall** be verified to ensure that they are indeed associated with the private key known by the purported owner of the public key. This is commonly known as proof-of-possession (POP). As a general rule, POP **should** be accomplished using the keys for their intended function. For example, it is recommended that the POP of a public key transport key be done using a key transport function, not a digital signature function.

7.2 Operational Phase

Keying material used during the cryptoperiod of a key is often stored for access as needed. During storage, the keying material **shall** be protected as specified in Section 5.2.2. During normal use, the keying material is stored either on the device or module that uses that material, or on a readily accessible storage media. When the keying material is required for operational use, the keying material is acquired from immediately accessible storage when not present in active memory within the device or module.

To provide continuity of operations when the keying material becomes unavailable for use from normal operational storage during its cryptoperiod (e.g., because the material is lost or corrupted), keying material may need to be recoverable. If an analysis of system operations indicates that the keying material needs to be recoverable, then the keying material **shall** either be backed up (see Section 7.2.2.1), or the system be designed to allow reconstruction (e.g., re-derivation) of the keying material. Acquiring the keying material from backup or by reconstruction is commonly known as key recovery (see Section 7.2.2.2).

At the end of a key's cryptoperiod, a new key needs to be available to replace the old key if operations are to be continued. This can be accomplished by rekeying (see Section 7.2.3.1), key update (see Section 7.2.3.2) or key derivation (see Section 7.2.3.3). A key **shall** be destroyed in accordance with Section 7.3.4 and **should** be destroyed as soon as that key is no longer needed in order to reduce the risk of exposure.

7.2.1 Normal Operational Storage

The objective of key management is to facilitate the operational availability of keying material for standard cryptographic purposes. Usually, a key remains operational until the end of the key's cryptoperiod (i.e., the expiration date). During normal operational use, keying material is available either in the device or module (e.g., in RAM) or in an immediately accessible storage media (e.g., on a local hard disk).

7.2.1.1 Device or Module Storage

Keying material may be stored in the device or module that adds, checks or removes the cryptographic protection on information. The storage of the keying material **shall** be consistent with Section 5, as well as with FIPS 140-2.

7.2.1.2 Immediately Accessible Storage Media

Keying material may need to be stored for normal cryptographic operations on an immediately accessible storage media (e.g., a local hard drive) during the cryptoperiod of the key. The storage requirements of Section 5.2.2 apply to this keying material.

7.2.2 Continuity of Operations

Keying material can become lost or unusable due to hardware damage, corruption or loss of program or data files, or system policy or configuration changes. In order to maintain the continuity of operations, it is often necessary for users and/or administrators to be able to recover keying materials from back-up storage. However, if operations can be continued without the backup of keying material (e.g., by key replacement), or the keying material can be recovered or reconstructed without being saved, it may be preferable not to save the keying material in order to lessen the possibility of a compromise of the keying material or other cryptographic related information.

The compromise of keying material affects continuity of operations (see Section 8.4). When keying material is compromised, the continuity of operations requires the establishment of entirely new key material (see Section 7.1.5), following an assessment of what keying material is affected and must be replaced.

7.2.2.1 Backup Storage

The backup of keying material on an independent, secure storage media provides a source for key recovery (see Section 7.2.2.2). Backup storage is used to store copies of information that is also currently available in normal operational storage during a key's cryptoperiod (i.e., in the cryptographic device or module, or on an immediately accessible storage media - see Section 7.2.1.1). Not all keys need be backed up. The storage requirements of Section 5.2.2 apply to keying material that is backed up. Tables 4 and 5 provide guidance about the backup of each type of keying material and other related information. An “OK” indicates that storage is permissible, but not necessarily required. The final determination for backup **should** be made based on the application in which the keying material is used. A detailed discussion about each type of key and other cryptographic information is provided in Appendix B.3.

Keying material maintained in backup **should** remain in storage for at least as long as the same keying material is maintained in storage for normal operational use (see Section 7.2.1). When no longer needed for normal operational use, the keying material and other related information **should** be removed from backup storage. When removed from backup storage, all traces of the information **shall** be destroyed in accordance with Section 7.3.4.

A discussion of backup and recovery is provided in [ITL Bulletin].

Table 4. Backup of keys.

Type of Key	Backup?
Private signature key	No (in general); non-repudiation would be in question. However, it may be warranted in some cases - a CA’s signing private key, for example. When required, any backed up keys must be stored under the owner’s control.
Public signature verification key	OK; its presence in a public-key certificate that is available elsewhere may be sufficient.
Symmetric authentication key	OK
Private authentication key	OK, if required by an application.
Public authentication key	OK; its presence in a public-key certificate that is available elsewhere may be sufficient.
Symmetric data encryption key	OK
Symmetric key wrapping key	OK
Random number generation key	Not necessary and may not be desirable, depending on the application.
Symmetric master key	OK
Private key transport key	OK
Public key transport key	OK; presence in a public-key certificate available elsewhere may be sufficient.
Symmetric key agreement key	OK
Private static key agreement key	No, unless needed for reconstruction during key recovery. However, when ephemeral information (e.g., a private ephemeral key agreement key) is used in a key agreement scheme, knowledge of the private static key agreement key and any public keys will not be sufficient.

Type of Key	Backup?
Public static key agreement key	OK; its presence in a public-key certificate that is available elsewhere may be sufficient.
Private ephemeral key agreement key	No
Public ephemeral key agreement key	No, unless needed for reconstruction during key recovery
Symmetric authorization key	OK
Private authorization key	OK
Public authorization key	OK; its presence in a public-key certificate that is available elsewhere may be sufficient.

Table 5. Backup of other cryptographic or related information.

Type of Keying Material	Backup?
Domain parameters	OK
Initialization vector	OK, if necessary
Shared secret	No
Secret seed	No
Public seed	OK, if required for the validation of domain parameters
Other public information	OK
Intermediate results	No
Key control information (e.g., IDs, purpose, etc.)	OK
Random number	OK
Passwords	OK
Audit information	OK

7.2.2.2 Key Recovery

Keying material that is in active memory or stored in normal operational storage may sometimes be lost or corrupted (e.g., from a system crash or power fluctuation). Some of the keying material is needed to continue operations and cannot easily be replaced. An assessment needs to be made of which keying material needs to be preserved for possible recovery at a later time.

The decision as to whether key recovery is required **should** be made on a case by case basis. The decision **should** be based on:

1. the type of key (e.g., private signature key, symmetric data encryption key),

2. the application in which the key will be used (e.g., interactive communications, file storage),
3. whether the key is "owned" by the local entity (e.g., a private key) or by another entity (e.g., the other entity's public key) or is shared (e.g., a symmetric data encyption key shared by two entities),
4. the role of the entity in a communication (e.g., sender of receiver), and
5. the algorithm or computation in which the key will be used (e.g., does the entity have the necessary information to perform a given computation if the key were to be recovered)[9].

The factors involved in a decision for or against key recovery **should** be carefully assessed. The trade-offs are concerned with continuity of operations versus the risk of possibly exposing the keying material and the information it protects if control of the keying material is lost. If it is determined that a key needs to be recovered, and the key is still active (i.e., the cryptoperiod of the key has not expired), then the key **shall** be replaced as soon as possible in order to limit its exposure (see Section 7.2.3).

Issues associated with key recovery and discussions about whether or not different types of cryptographic material need to be recoverable are provided in Appendix B.

7.2.3 Key Replacement

There are several reasons for replacing a key with another key.

1. The key may have been compromised.
2. The key's cryptoperiod may be nearing expiration.
3. It may be desirable to limit the amount of data protected with any given key.

A key may be replaced by rekeying or by key update.

7.2.3.1 Rekeying

If the new key is generated in a manner that is entirely independent of the "value" of the old key, the process is known as rekeying. This replacement **shall** be accomplished using one of the key establishment methods discussed in Section 7.1.5. Rekeying is used when a key has been compromised (provided that the rekeying scheme itself is not compromised) or when the cryptoperiod is nearing expiration.

7.2.3.2 Key Update

If the "value" of the new key is dependent on the value of the old key, the process is known as key update (i.e., the current key is modified to create a new key). This **should** be accomplished by applying a non-reversible function to the old key and other data. Unlike rekeying, key update does not require the exchange of any new information between the entities that previously shared the old key. For example, the two entities may agree to update their shared key on the first day of each month. Since a non-reversible function is used in the update process, previous keys are protected in the event that a key is compromised. However, future keys are not protected. After

[9] This could be the case when performing a key establishment process for some key establishment schemes (see SP 800-56).

a limited number of updates, new keying material **should** be established by employing a fresh rekey operation (see Section 7.2.3.1). Key update is often used to limit the amount of data protected by a single key, but it would not be used to replace a compromised key.

7.2.4 Key Derivation

Symmetric and private cryptographic keys may be derived from other secret values, sometimes called master keys. The secret values and possibly other information are input into a function that outputs one or more derived keys. In contrast to key replacement, the derived keys are often used for new purposes, rather than for replacing the secret values from which they are derived. The derivation function **should** be a non-reversible function so that the secret values cannot be determined from the derived keys. In addition, it **should not** be possible to determine a derived key from other derived keys. It should be noted that the strength of the derived key is no greater than the strength of the secret values from which the key was derived.

Four of the cases are discussed below.

1. *Two parties derive common keys from a common shared secret.* This approach is used in the key agreement techniques specified in [SP 800-56]. The security of this process is dependent on the security of the shared secret and the specific key derivation function used. If the shared secret is known, the derived keys may be determined. A key derivation function specified in [SP 800-56] **shall** be used for this purpose. These derived keys may be used to provide the same confidentiality, authentication, and data integrity services as randomly generated keys

2. *Individual entity keys are derived from a master key.* This is often accomplished by using the master key, entity ID, and other known information as input to a function that generates the entity keys. The security of this process depends upon the security of the master key and the key derivation function. If one of the entities knows the master key, the other entity keys may all be generated. Therefore, keys derived from a master key are only as secure as the master key itself. As long as the master key is kept secret, these keys may be used in the same manner as randomly generated keys.

3. *The individual entity key is derived from a master key and the entity password.* These secret values are input to the key derivation function along with other known information. The security of a derived entity key is dependent upon the security of the master key, the security of the password, and the strength of the key derivation process. This form of key derivation is often used to add the entity authentication service to the derived keys. As long as the secret inputs are kept secret, these keys may be used in the same manner as randomly generated keys.

4. *The individual entity key is derived from the entity password*[10]. This is accomplished using a password, entity ID, and other known information as input to the key derivation function. This technique differs from the previous technique in that no master key is used. Therefore, the security of the process depends solely upon the security of the password and the key derivation process. If the entity password is known or can be guessed, then the corresponding derived entity key may be generated. Since the number

[10] This technique is used in RSA PKCS #5.

of possible (or likely) passwords is often significantly smaller than the number of possible keys, keys derived in this manner are likely to be less secure than randomly generated keys. Keys derived in this case **shall** be used for authentication purposes only and not for general encryption. Other uses of the keys have not been evaluated (see Appendix C).

7.3 Post-Operational Phase

During the post-operational phase, keying material is no longer in operational use, but access to the keying material is still possible.

7.3.1 Archive Storage and Key Recovery

If the keying material needs to be recoverable after the end of its cryptoperiod, the keying material **shall** be either archived or the system be designed to allow reconstruction (e.g., re-derivation) of the keying material. Acquiring the keying material from archive storage, or by reconstruction is commonly known as key recovery.

A key management archive is a repository containing keying material and other related information of historical interest. Not all keying material needs to be archived. An organization's security plan **should** indicate the types of information that are to be archived (see Part 2).

While in storage, archived information may be either static (i.e., never changing) or may need to be re-encrypted under a new archive encryption key. Archived data **should** be stored separately from operational data, and multiple copies of archived cryptographic information **should** be provided in physically separate locations (i.e., it is recommended that the key management archive be backed up). For critical information encrypted under archived keys, it may be necessary to back up archive keys and to store multiple copies of archived keys in separate locations.

When archived, keying material **should** be archived prior to the end of the validity period (i.e., cryptoperiod, reliance period) of the key. When no longer required, the keying material **shall** be destroyed in accordance with Section 7.3.4.

Archived cryptographic information requires protection in accordance with Section 5.2.2. Confidentiality is provided by an archive encryption key (one or more encryption keys that are used exclusively for the encryption of archived information), by another key that has been archived, or by a key that may be derived from an archived key. When encrypted by the archive encryption key, the encrypted keying material **shall** be re-encrypted by any new archive encryption key at the end of the cryptoperiod of the old archive encryption key. When the keying material is re-encrypted, integrity values on that keying material **shall** be recomputed.

Likewise, integrity protection is provided by an archive integrity key (one or more authentication or digital signature keys that are used exclusively for the archive) or by another key that has been archived. At the end of the cryptoperiod of the archive integrity key, new integrity values **shall** be computed on the archived information on which the old archive integrity key was applied.

The archive keys may be either symmetric keys, or public key pairs. The keys used for confidentiality and integrity **shall** be different, and **shall** be protected in the same manner as their key type (see Section 5).

The key archiving technique that is used **should** be appropriate to the storage method used for the encrypted information protected under archived keys. For example, the differences between the organization of random access data files and database management system (DBMS) files are such that a method used to store keys under which random access files are protected and to associate the keys with the protected data would likely be both inappropriate and inadequate for DBMS applications. (Assume that the elements stored in a database are created at different times and protected under different keys. The key information would then need to be associated with data elements rather than data files.)

Tables 6 and 7 indicate the desirability of archiving the various cryptographic information. An "OK" in column 2 (Archive?) indicates that archival is permissible, but not necessarily required. Column 3 (Retention period) indicates the minimum time that the key **should** be retained in the archive. Additional advice on the storage of keying material in archive storage is provided in Appendix B.3.

Table 6: Archive of keys.

Type of Key	Archive?	Retention period (minimum)
Private signature key	No	
Public signature verification key	OK	Until no longer required to verify data signed with the assoc. private key
Symmetric authentication key	OK	Until no longer needed to authenticate data.
Private authentication key	No	
Public authentication key	OK	Until no longer required to verify the authenticity of data that was authenticated with the assoc. private key
Symmetric data encryption key	OK	Until no longer needed to decrypt data encrypted by this key
Symmetric key wrapping key	OK	Until no longer needed to decrypt keys encrypted by this key
Symmetric random number generator key	No	
Symmetric master key	OK, if needed to derive other keys for archived data	Until no longer needed to derive other keys
Private key transport key	OK	Until no longer needed to decrypt keys encrypted by this key

Type of Key	Archive?	Retention period (minimum)
Public key transport key	No	
Symmetric key agreement key	OK	
Private static key agreement key	OK if needed to reconstruct keying material	Until no longer needed to reconstruct keying material
Public static key agreement key	OK if needed to reconstruct keying material	Until no longer needed to reconstruct keying material
Private ephemeral key agreement key	No	
Public ephemeral key agreement key	No	
Symmetric authorization key	No	
Private authorization key	No	
Public authoriza tion key	No	

Table 7: Archive of Other Cryptographic and Related Information.

Type of Key	Archive?	Retention period (minimum)
Domain parameters	OK	Until all keying material, signatures and signed data using the domain parameters are removed from the archive
Initialization vector	OK; normally stored with the protected information	Until no longer needed to process the protected data
Shared secret	No, unless needed to validate or reconstruct derived keying material for archived information	Until no longe r needed to validate or reconstruct derived keying material for archived information.
Secret seed	No	
Public seed	OK	Until no longer needed to process generated data

Type of Key	Archive?	Retention period (minimum)
Other public information	OK	Until no longer needed to process data using the public information
Intermediate result	No	
Key control information (e.g., IDs, purpose)	OK	Until the associated key is removed from the archive
Random number	No	
Password	No, unless used to detect the reuse of old passwords	Until no longer needed to detect password reuse
Audit information	OK	Until no longer needed

After the end of a key's cryptoperiod, keying material may be recovered from archival storage, providing that the keying material has been archived. Alternatively, the keying material may be reconstructed (e.g., rederived), if the key management system has been appropriately designed.

Key recovery of archived keying material may be required to remove (e.g., decrypt) or check (e.g., verify a digital signature or a MAC) the cryptographic protections on archived data. The key recovery process results in retrieving the desired keying material from archive storage and placing it in active memory or normal operational storage in order to perform the required cryptographic operation. Immediately after completing this operation, the keying material **shall** be erased from the active memory and normal operational storage (see Section 7.3.4). Further advice on key recovery issues is provided in Appendix B.

7.3.2 User De-registration

When an entity ceases to be a member of a security domain, the entity **shall** be de-registered. De-registration is intended to prevent other entities from relying on or using the de-registered entity's keying material.

All records of the entity and the entity's associations **shall** be marked to indicate the entity is no longer a member of the security domain, but the records **should not** be deleted. To reduce confusion and unavoidable human errors, identification information associated with the de-registered entity **should not** be re-used (at least for a period of time). For example, if a "John Wilson" retires and is de-registered on Friday, it is recommended that the identification information assigned to his son "John Wilson", who is hired the following Monday, be different.

7.3.3 Key De-registration

Registered keying material may be associated with the identity of a key owner, owner attributes (e.g., email address), or role or authorization information. When the keying material is no longer needed, or the associated information becomes invalid, the keying material **should** be de-registered (i.e., all records of the keying material and its associations **should** be marked to indicate that the key is no longer in use).

When a cryptographic key is compromised, that key and any associated keying material **should** be de-registered. For example, when a private key is compromised, it is recommended that the corresponding public key certificate be revoked. Certificate revocation because of a key compromise advertises that the binding between the owner and the key is no longer to be trusted. In a PKI, key de-registration is commonly achieved by including the certificate in a list of revoked certificates (i.e., a CRL). Where the PKI uses online status mechanisms (e.g., the Online Certificate Status Protocol [OCSP]), de-registration is achieved by informing the appropriate certificate status server(s).

Keying material **should** be de-registered when the attributes associated with an entity are modified. For example, if an entity's email address is associated with a public key, and the entity's address changes, the keying material **should** be de-registered to indicate that the associated attributes have become invalid. Unlike the case of key compromise, the entity could safely re-register the public key after modifying the entity's attributes through the user registration process (see Section 7.1.1).

7.3.4 Key Destruction

All copies of the private or symmetric key **shall** be destroyed as soon as no longer required (e.g., for archival or reconstruction activity) in order to minimize the risk of a compromise. Any media on which unencrypted keying material requiring confidentiality protection is stored **shall** be erased in a manner that removes all traces of the keying material so that it cannot be recovered by either physical or electronic means[11]. Public keys may be retained or destroyed, if desired.

7.3.5 Key Revocation

It is sometimes necessary to remove keying material from use prior to the end of its normal cryptoperiod for reasons that include key compromise, removal of an entity from an organization, etc. This process is known as key revocation and is used to explicitly revoke a symmetric key or the public key of a key pair, although the private key associated with public key is also revoked.

Key revocation may be accomplished using a notification indicating that the continued use of the keying material is no longer recommended. The notification could be provided by actively sending the notification to all entities that might be using the revoked keying material, or by allowing the entities to request the status of the keying material (i.e, a “push” or a “pull” of the status information). The notification **should** include a complete identification of the keying material, the date and time of revocation and the reason for revocation, when appropriate (e.g., key compromised). Based on the revocation information provided, other entities could then make a determination of how they would treat information protected by the revoked keying material.

For example, if a signature verification public key is revoked because an entity left an organization, it may be appropriate to honor all signatures created prior to the revocation date. If a signing private key is compromised, an assessment needs to be made as to whether or not information signed prior to the revocation would be considered as valid.

[11] A simple deletion of the keying material might not completely obliterate the information. Erasing the information might require overwriting that information with other non-related information, such as random bits, or all zero or one bits.

As another example, a symmetric key that is used to generate MACs may be revoked so that it is not used to generate MACs on new information. However, the key may be retained so that archived documents can be verified.

The details for key revocation **should** reflect the lifecycle for each particular key. If a key is used in a pair-wise situation (e.g., two entities communicating in a secure session), the entity revoking the key **shall** inform the other entity. If the key has been registered with an infrastructure, the entity revoking the key cannot always directly inform the other entities that may rely upon that key. Instead, the entity revoking the key **shall** inform the infrastructure that the key **shall** be revoked (e.g., using a certificate revocation request). The infrastructure **shall** respond by de-registering the key material (see 7.3.3).

7.4 Obsolete/Destroyed Phase

The keying material is no longer available. All records of its existence may have been deleted. However, some organizations may require the retention of key management records for audit purposes. For example, if a copy of an ostensibly destroyed key is found in an uncontrolled environment, records of the identity of the key and its use may be helpful in determining what was protected under the key, the probability that the information or process was compromised, and how to recover from any assumed compromise.

8 GENERAL KEY MANAGEMENT GUIDANCE

This section is intended to provide general guidance concerning key usage, cryptoperiod length, assurance of domain parameter and public key validity, key compromise, accountability, audit and key management system survivability. In addition, guidance is provided for selecting appropriate algorithms and key sizes and using the key establishment techniques specified in [SP 800-56].

8. 1 Key Usage

It is a general precept of secure key management that a single key **shall** be used for only one purpose (e.g., encryption, authentication, key wrapping, random number generation, or digital signatures). There are several reasons for this:

1. Each use of a key may expose the key to attack in some way, and using the key for different cryptographic processes may expose the key to additional attacks.
2. Limiting the use of a key limits the damage that could be done if the key is compromised.
3. Some uses of keys interfere with each other. For example, consider a key pair used for both key transport and digital signatures. A private key transport key that is used to decrypt a symmetric data encryption key that is, in turn, used for the decryption of an encrypted file may need to be retained for use as long as the data in the encrypted file needs to be accessed. The retention period may be longer than the cryptoperiod of the associated public key transport key that was used to encrypt the symmetric data encryption key. On the other hand, a public signature verification key used to validate digital signatures needs to be retained as long as the digital signature on any information signed by the associated private signature key may need to be verified. However, the associated private signature key **shall** be destroyed at the expiration of its validity period to prevent its compromise. In this example, the longevity requirements for the key transport key pairs and the digital signature key pairs contradict each other.

This principle does not preclude using a single key in cases where the same process can provide multiple services. This is the case, for example, when a digital signature provides non-repudiation, authentication and integrity protection using a single digital signature, or when a single symmetric data encryption key can be used to encrypt and authenticate data in a single cryptographic operation (e.g., using an authenticated encryption operation, as opposed to separate encryption and authentication operations). Also refer to Section 3.7.

8.2 Cryptoperiods

A cryptoperiod is the time span during which a specific key is authorized for use by legitimate entities, or the keys for a given system will remain in effect. A suitably defined cryptoperiod:

1. limits the amount of information protected by a given key that is available for cryptanalysis,
2. limits the amount of exposure if a single key is compromised,
3. limits the use of a particular algorithm to its estimated effective lifetime,

4. limits the time available for attempts to penetrate physical, procedural, and logical access mechanisms that protect a key from unauthorized disclosure
5 limits the period within which information may be compromised by inadvertent disclosure of keying material to unauthorized entities, and
6. limits the time available for computationally intensive cryptanalytic attacks (in applications where long-term key protection is not required).

Trade-offs associated with the determination of cryptoperiods involve the risk and consequences of exposure.

8.2.1 Risk Factors Affecting Cryptoperiods

Among the factors affecting the risk of exposure are:

1. the strength of the cryptographic mechanisms (e.g., the algorithm, key length, mode of operation),
2. the embodiment of the mechanisms (e.g., FIPS 140-2 Level 4 implementation, or software implementation on a Microsoft Windows machine),
3. the operating environment (e.g., secure limited access facility, open office environment, or publicly accessible terminal),
4. the volume of information flow or the number of transactions,
5. the security life of the data,
6. the security function (e.g., data encryption, digital signature, key production or derivation, key protection),
7. the rekeying method (e.g., keyboard entry, rekeying using a key loading device where humans have no direct access to key information, remote rekeying within a PKI),
8. the key update or key derivation process,
9. the number of nodes in a network that share a common key,
10. the number of copies of a key and the distribution of those copies, and
11. the threat to the information (e.g., who the information is protected from, and what are their perceived technical capabilities and financial resources to mount an attack).

In some cases, increased risk might suggest shorter cryptoperiods, while in other cases, increased risk might suggest a need for longer cryptoperiods. For example, some cryptographic algorithms might be more vulnerable to cryptanalysis if the adversary has access to a large volume of stereotyped data that is encrypted under the same key. Particularly where a symmetric key is shared among several entities, it may be prudent to employ short cryptoperiods for such algorithms. On the other hand, where manual key distribution methods are subject to human error and frailty, more frequent key changes might actually increase the risk of exposure. In these cases, especially when very strong cryptography is employed, it may be more prudent to have fewer, well-controlled manual key distributions rather than more frequent, poorly controlled manual key distributions.

In general, where strong cryptography is employed, physical, procedural, and logical access protection considerations often have more impact on cryptoperiod selection than do algorithm and key size factors. In the case of FIPS-approved algorithms, modes of operation, and key sizes, adversaries may be able to access keys through penetration or subversion of a system with less expenditure of time and resources than would be required to mount and execute a cryptographic attack.

8.2.2 Consequence Factors Affecting Cryptoperiods

The consequences of exposure are measured by the sensitivity of the information, the criticality of the processes protected by the cryptography, and the cost of recovery from the compromise of the information or processes. Sensitivity refers to the lifespan of the information being protected (e.g., 10 minutes, 10 days or 10 years) and the potential consequences of a loss of protection for that information (e.g., the disclosure of the information to unauthorized entities). In general, as the sensitivity of the information or the criticality of the processes protected by cryptography increase, the length of the associated cryptoperiods **should** decrease in order to limit the damage that might result from each compromise. This is, of course, subject to the caveat regarding the security and integrity of the rekeying , key update or key derivation process (see Sections 7.2.3 and 7.2.4). Particularly where denial of service is the paramount concern, and there is a significant potential for error in the rekeying, key update or key derivation process, short cryptoperiods may be counterproductive.

8.2.3 Other Factors Affecting Cryptoperiods

8.2.3.1 Communications Versus Storage

Keys that are used for confidentiality protection of communications exchanges in which the information being exchanged does not require long term protection may often have shorter cryptoperiods than keys used for the protection of stored data.

8.2.3.2 Cost of Key Revocation and Replacement

In some cases, the costs associated with changing keys are painfully high. Examples include decryption and subsequent re-encryption of very large databases, decryption and re-encryption of distributed databases, and revocation and replacement of a very large number of keys (e.g., where there are very large numbers of geographically and organizationally distributed key holders). In such cases, the expense of security measures necessary to support longer cryptoperiods may be justified (e.g., costly and inconvenient physical, procedural, and logical access security; use of cryptography strong enough to support longer cryptoperiods even where this may result in significant additional processing overhead. Alternatively, the cryptoperiod may be shorter than would otherwise be necessary in order to limit the size of Certificate Revocation Lists (CRLs) or Compromised Key Lists (CKLs).

8.2.4 Usage Periods Versus Cryptoperiod

In some cases, a key may have different cryptoperiods for its use in originating cryptographic protection and the subsequent processing of protected information by a recipient.

8.2.4.1 Public Key and Private Key Cryptoperiods

For key pairs, each key of the pair has its own cryptoperiod. That is, each key is used by an "originator" to apply cryptographic protection (e.g., create a digital signature) or by a "recipient" to subsequently process the protected information (e.g., verify a digital signature), but not both. Examples of cryptoperiod issues associated with public key cryptography include:

1. The cryptoperiod of a private key transport key may be longer than the cryptoperiod of the associated public key (i.e., the public key transport key). The public key is used for a fixed period of time to encrypt keying material. That period of time may be indicated by the *expiration date* on a public key certificate. The private key will need to be retained as long as there is a need to recover (i.e., decrypt) the key(s) encrypted by the public key (often after the public key has been destroyed.
2. In contrast, the cryptoperiod of a private authentication key that is used to sign challenge information is basically the same as the cryptoperiod of the associated public key (i.e., the public authentication key). That is, when the private key will not be used to sign challenges, the public key is no longer needed.
3. If a private signature key is used to generate digital signatures as a proof-of-origin in the future, the cryptoperiod of the private key is significantly shorter than the cryptoperiod of the associated public signature verification key. In this case, the private key is usually intended for use for a fixed period of time, after which time the key owner **shall** destroy the private key. The public key **should** be available for the longest period of time that the signature may need to be verified. In this case, the effective cryptoperiod for the public key (which may end before all necessary verifications of the digital signature) may be extended by supplementing the strength of the protection mechanism (e.g., the digital signature on the certificate) by submitting the signed information to a trusted electronic archive, or by obtaining a cryptographic timestamp on the signed information. This would provide assurance that the information was signed while the private signature key was still valid.

8.2.4.2 Symmetric Key Cryptoperiods

For symmetric keys, a single key is used for both applying the protection (e.g., encrypting or computing a MAC) and processing the protected information (e.g., decrypting the encrypted information or verifying a MAC). The cryptoperiod of the key consists of an "originator usage period" and a "recipient usage period". The originator usage period pertains to the time during which the cryptographic protection is originally applied (e.g., the information is originally encrypted, or the MAC is originally computed); the key **shall not** be used to apply this protection at a later time (e.g., the key must not be used for encrypting information or computing the original MAC after the end of the originator usage period). The recipient usage period pertains to the time during which the key can be used for the subsequent processing of the protected information (e.g., decrypting the encrypted information or verifying the MAC). The usage periods begin at the beginning of the cryptoperiod; the recipient usage period may extend beyond the originator usage period; the recipient usage period ends at the end of the cryptoperiod. In many cases, the originator and recipient usage periods are the same. See Figure 3.

Examples of the use of the usage periods include:

a. When a symmetric key is used only for securing communications, the difference between the period of time from the originator's application of protection to the recipient's processing is negligible. In this case, the key is authorized for either purpose during the entire cryptoperiod, i.e., the originator usage period and the recipient usage period are the same.

b. When a symmetric key is used to protect stored information, the originator usage period (when the originator applies cryptographic protection to stored information) may end much earlier than the recipient usage period (when the stored information is processed). In this case, the cryptoperiod begins at the initial time authorized for the application of protection with the key, and ends with the latest time authorized for processing using that key. In general, the recipient usage period for stored information will continue beyond the originator usage period, so that the stored information may be authenticated or decrypted at a later time.

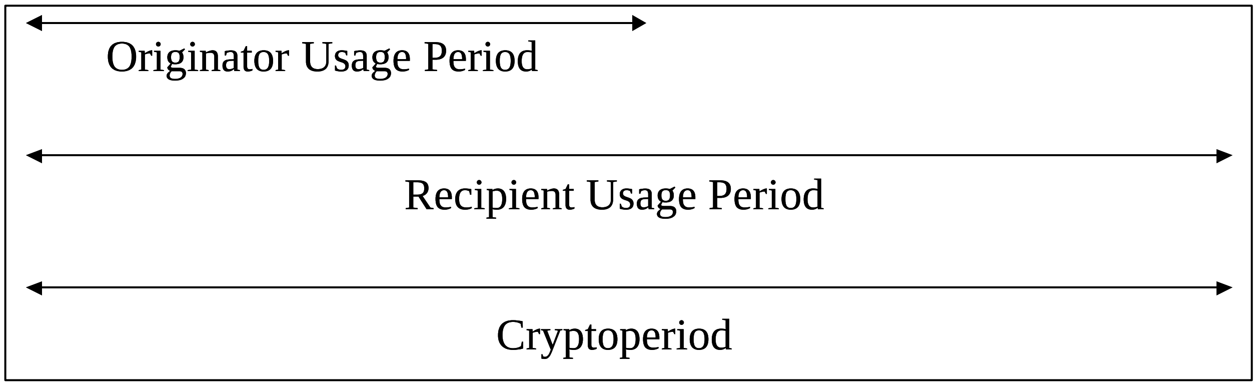

Figure 3: Symmetric Key Cryptoperiod

8.2.5 Cryptoperiod Recommendations for Specific Key Types

The cryptoperiod required for a given key may be affected by key type as much as by the usage environment and data characteristics described above. Some general cryptoperiod guidelines for various key types are suggested below. Note that the cryptoperiods suggested are only rough order of magnitude guidelines. Most are on the order of 1-2 years, based on 1) a desire for maximum operational efficiency and 2) assumptions regarding minimum criteria regarding usage environment (see [FIPS 140-2], [SP 800-14], [SP 800-21], and [SP 800-37]). The factors described in Paragraphs 8.2.1 through 8.2.4 **should** be used to determine actual cryptoperiods for specific usage environments,

1. *Private signature key*:

   a. Type Considerations: In general, the cryptoperiod of a private signature key may be shorter than the cryptoperiod of the corresponding public signature verification key.

   b. Cryptoperiod: Given the use of FIPS-approved algorithms and key sizes, and an expectation that the security of the key storage and use environment will increase as the sensitivity and/or criticality of the processes for which the key provides integrity protection increases, a maximum cryptoperiod of about 1-3 years is recommended.

2. *Public signature verification key*:

   a. Type Considerations: In general, the cryptoperiod of a public signature verification key may be longer than the cryptoperiod of the corresponding private signature key. The cryptoperiod is, in effect, the period during which any signature computed using the associated private signature key needs to be verified. A long cryptoperiod for the public signature verification key poses a relatively minimal security concern. If the private key is compromised, and an adversary achieves access to the signed data, it may be possible for an adversary to modify signed data and recalculate the signature. Note that this requires either a cryptographic break or illicit access to the private key of the signer <u>and</u> write access to the signed data. For further advice, see Appendix C.

b. Cryptoperiod: The cryptoperiod may be on the order of tens of years, though due to the long exposure of protection mechanisms to hostile attack, reliability of the signature is reduced with the passage of time. (That is, for any given algorithm and key size, vulnerability to cryptanalysis is expected to increase with time. Although choosing the strongest available algorithm and a large key size can minimize this vulnerability to cryptanalysis, exposure to attacks on physical, procedural, and logical access control mechanisms is not affected.)

3. *Symmetric authentication key*:

   a. Type Considerations: The cryptoperiod of a secret authentication key depends on the sensitivity of the type of information it protects and the protection afforded the key. For very sensitive information (e.g., information that one would not like to be compromised (unprotected) if the key used to protect other information were compromised), the authentication key may need to be unique to the protected information. Otherwise, suitable cryptoperiods may extend beyond a single use. The originator usage period of a secret authentication key applies to the use of that key in applying the original cryptographic protection for the information (e.g., computing the MAC to be associated with the authenticated information); new MACs **shall not** be computed on information after the end of the originator usage period. However, the key **should** be available to verify the MAC on the protected data for as long as the protection is required, i.e., the recipient usage period of the key, when used only for verification purposes, is the lifetime of the protected information. Note, however, that cryptoperiods are normally pre-determined and may not be adequate for the lifetime of the protected data. If the key is compromised, and given access to the authenticated data, it may be possible for an adversary to modify authenticated data and recalculate the MAC. Note that this requires either a cryptographic break or illicit access to the secret key of the signer <u>and</u> write access to the authenticated data.

   b. Cryptoperiod: Given the use of FIPS-approved algorithms and key sizes and an expectation that the security of the key storage and use environment will increase as the sensitivity and/or criticality of the processes for which the key provides integrity protection increases, a maximum cryptoperiod (i.e., recipient usage period) of up to 2 years is recommended.

4. *Private authentication key*:

   a. Type Considerations: A private authentication key may be used multiple times. Its associated public key could be certified, for example, by a Certificate Authority. In most cases, the cryptoperiod of the authentication private key is the same as the cryptoperiod of the associated public key.

   b. Cryptoperiod: An appropriate cryptoperiod for the private authentication key would be 1-2 years, depending on its usage environment and the sensitivity/criticality of the authenticated information.

5. *Public authentication key*:

a. Type Considerations: In most cases, the cryptoperiod of a public authentication key is the same as the cryptoperiod of the associated private authentication key. The cryptoperiod is, in effect, the period during which the authentication of information protected by the associated authentication private key needs to be verified.

b. Cryptoperiod: An appropriate cryptoperiod for the authentication public key would be 1-2 years, depending on its usage environment and the sensitivity/ criticality of the authenticated information.

6. *Symmetric data encryption key*:

a. Type Considerations: A symmetric data encryption key is used to protect stored data, messages or communications sessions. Based primarily on the consequences of compromise, a data encryption key that is used to encrypt large volumes of information over a short period of time (e.g., for a link encryption) **should** have a relatively short cryptoperiod. An encryption key used to encrypt less information could have a longer cryptoperiod. The originator usage period of a symmetric data encryption key applies to the use of that key in applying the original cryptographic protection for information (i.e., encrypting the information) (see Section 8.2.4).

During the originator usage period, information may be encrypted by the data encryption key; the key **shall not** be used for performing an encryption operation on information beyond this period. However, the key **should** be available to decrypt the encrypted information for as long as the encrypted information exists, i.e., the recipient usage period of the key, when used only for decryption, is the lifetime of the encrypted information.

b. Cryptoperiod: The cryptoperiod recommended for the encryption of large volumes of information over a short period of time (e.g., for a link encryption) is on the order of a day or a week. An encryption key used to encrypt smaller volumes of information might have a cryptoperiod of up to one month. In the case of symmetric data encryption keys used to encrypt single messages or single communications sessions, the cryptoperiod may be based on availability of the key to decrypt the encrypted information for as long as the encrypted information exists. This may be on the order of tens of years, though confidence in the confidentiality of the information is reduced with the passage of time.

7. *Symmetric key wrapping key*:

a. Type Considerations: A symmetric key wrapping key that is used to encrypt very large numbers of keys over a short period of time **should** have a relatively short cryptoperiod. If a small number of keys are encrypted, the cryptoperiod of the key wrapping key could be longer. The originator usage period of a symmetric key wrapping key applies to the use of that key in providing the original protection for information (i.e., encrypting the key that is to remain secret); keys **shall not** be encrypted using the key wrapping key after the end of the originator usage period. However, the key wrapping key **should** be available to decrypt the encrypted key for as long as the encrypted key exists in its encrypted form, i.e., the recipient usage period of the key wrapping key, when used only for decryption, is the lifetime of the encrypted key.

Some symmetric key wrapping keys are used for only a single message or communications session. In the case of these very short-term key wrapping keys, an appropriate cryptoperiod is a single communication session. It is assumed that the key as encrypted by the key wrapping key will not be retained in its encrypted form, so the cryptoperiod of the key wrapping key as used for encryption (the originator usage period) is the same as that same key used for decryption (the recipient usage period).

b. Cryptoperiod: The recommended cryptoperiod for a symmetric key wrapping key that is used to encrypt very large numbers of keys over a short period of time is on the order of a day or a week. If a relatively small number of keys are to be encrypted under the key wrapping key, the cryptoperiod of the key wrapping key could be up to a month. In the case of keys used for only a single message or communications session, the cryptoperiod would be limited to a single communication session.

8. *Symmetric RNG key*:

a. Type Considerations: Symmetric RNG keys are used in deterministic random number generation functions. The cryptoperiod of a key used to create random numbers depends on the amount of its use, though RNG keys may be used for an extended period of time. The key needs to be changed before the generation function begins to degenerate or cycle. Depending on the nature of the random number generation function, and how the output is used to generate keying material, the need to rekey may vary considerably. Unlike the case for many other key types, deterministic mathematical considerations may drive symmetric RNG key cryptoperiod selection more than factors related to the security environment and procedures. [This text may, and probably will, change based on X9F1 discussions re the RNG standard.]

b. Cryptoperiod: Suitable cryptoperiods might range from a month to two years. [Hopefully, the RNG standard under development (ANSI X9.82) will allow us to provide better advice.]

9. *Symmetric master key*:

a. Type Considerations: A symmetric master key may be used multiple times to derive other keys. Therefore, a suitable cryptoperiod depends on the nature and use of the keys derived from the master key and on considerations provided earlier in Section 8.2. The cryptoperiod of a key derived from a master key could be relatively short, e.g., a single use or a communication session or transaction. In such cases, the period of the master key can be determined based on factors described for symmetric data encryption keys.

Alternatively, keys derived from the same master key could be used for entirely different purposes (e.g., one key for encryption, another key for HMAC computations). The originator usage period of a derived key applies to the use of that key in applying the original cryptographic protection for information (e.g., encrypting the information). The derived key may need to be available to process the protected information for as long as the protected information exists (i.e., the recipient usage period of the derived key, when used only for the subsequent processing of the protected information, is the lifetime of the protected information). For example, if a key is derived to encrypt information for a single communication session, but the encrypted information is to be retained for a year, the originator usage period of the key when used for encryption is the length of the communication session. The recipient usage period of the key (i.e., the cryptoperiod), as used for decryption of the encrypted information, is one year. If operational procedures require retention of the master key to reconstruct such derived keys, the extended cryptoperiod would necessarily apply to the master key as well.

b. Cryptoperiod: Subject to the variables described above, an appropriate cryptoperiod for the symmetric master key might be 1 year, depending on its usage environment and the sensitivity/ criticality of the authenticated information.

10. *Private key transport key*:

a. Type Considerations: A private key transport key may be used multiple times. Due to the potential need to decrypt keys some time after they have been encrypted for transport, the cryptoperiod of the private key transport key may be longer than the cryptoperiod of the associated public key. The cryptoperiod of the private key is the length of time during which any keys encrypted by the associated key transport public key need to be decrypted.

b. Cryptoperiod: Given 1) the use of FIPS-approved algorithms and key sizes, 2) the volume of information that may be protected by keys encrypted under the associated public transport key, and 3) an expectation that the security of the key storage and use environment will increase as the sensitivity and/or criticality of the processes for which the key provides protection increases; a maximum cryptoperiod of about 2 years is recommended.

11. *Public key transport key*:

a. Type Considerations: The cryptoperiod for the public key transport key is that period of time during which the public key may be used to actually apply the encryption operation to the keys that will be protected. Public key transport keys can be public knowledge. Since the underlying plaintext keys tend to be random, plaintext assuming attacks are relatively difficult to execute. The driving factor in establishing the public key transport key cryptoperiod is the cryptoperiod of the associated private key transport key. As indicated in the private key transport key discussion, due to the potential need to decrypt keys some time after they have been encrypted for transport, the cryptoperiod of the public key transport key may be shorter than that of the associated private key.

b. Cryptoperiod: Based on cryptoperiod assumptions for associated private keys, a maximum recommended cryptoperiod might be about 1 - 2 years.

12. *Symmetric key agreement key*:

a. Type Considerations: A symmetric key agreement key may be used multiple times. The cryptoperiod of these keys depend on 1) environmental security factors, 2) the nature (e.g., types and formats) and volume of keys they are used to establish, and 3) the details of the key agreement algorithms and protocols employed. Note that symmetric key agreement keys may be used to establish symmetric keys (e.g., symmetric data encryption keys) or other keying material (e.g., IVs).

b. Cryptoperiod: Given an assumption that the cryptography that employs symmetric key agreement keys 1) employs an algorithm and key scheme compliant with NIST standards, 2) the cryptographic device meets FIPS 140-2 requirements, and 3) the security levels are established in conformance to SP 800-37, an appropriate cryptoperiod for the key would be 1-2 years.

13. *Private static key agreement key*:

    a. Type Considerations: A private static key agreement key may be used multiple times. As in the case of symmetric key agreement keys, the cryptoperiod of these keys depend on 1) environmental security factors, 2) the nature (e.g., types and formats) and volume of keys they are used to establish, and 3) the details of the key agreement algorithms and protocols employed. Note that private static key agreement keys may be used to establish symmetric keys (e.g., key wrapping keys) or other keying material (e.g., IVs).

    b. Cryptoperiod: The cryptoperiod of the private static key agreement key and its associated public key would be the same, (e.g., for the cryptoperiod of the certificate). Given an assumption that the cryptography that employs private static key agreement keys 1) employs an algorithm and key scheme compliant with NIST standards, 2) the cryptographic device meets FIPS 140-2 requirements, and 3) the security levels are established in conformance to SP 800-37, an appropriate cryptoperiod for the key would be 1-2 years.

14. *Public static key agreement key*:

    a. Type Considerations: The cryptoperiod for a public static key agreement key is the same as the cryptoperiod of the associated private static key agreement key. See the discussion for the private static key agreement key. However, the length of the public static key agreement key cryptoperiod is of far less concern, from a security point of view, than is that of the associated private key.

    b. Cryptoperiod: The cryptoperiod of the public static key agreement key is the same as the associated private static key agreement key.

15. *Private ephemeral key agreement key*:

    a. Type Considerations: Private ephemeral key agreement keys are the private key elements of asymmetric key pairs that are used only once to establish one or more keys. Private ephemeral key agreement keys may be used to establish symmetric keys (e.g., key wrapping keys)or other keying material (e.g., IVs).

    b. Cryptoperiod: Private ephemeral key agreement keys are one-time use keys. The cryptoperiod of a private ephemeral key agreement key is the duration of a single key agreement process.

16. *Public ephemeral key agreement key*:

    a. Type Considerations: Public ephemeral key agreement keys are the public key elements of asymmetric key pairs that are used only once to establish one or more keys.
    b. Cryptoperiod: Public ephemeral key agreement keys are one-time use keys. The cryptoperiod of a public ephemeral key agreement key is the duration of a single key agreement process.

17. *Symmetric authorization key*:

    a. Type Considerations: A symmetric authorization key may be used for an extended period of time, depending on the resources that are protected and the role of the entity authorized for access. Primary considerations in establishing the cryptoperiod for symmetric authorization keys include the robustness of the key, the adequacy of the cryptographic method, and the adequacy of key protection mechanisms and procedures.

b. Cryptoperiod: Given the use of FIPS-approved algorithms and key sizes, and an expectation that the security of the key storage and use environment will increase as the sensitivity and criticality of the authorization processes increases, it is recommended that cryptoperiods be less than two years.

18. *Private authorization key*:

a. Type Considerations: A private authorization key may be used for an extended period of time, depending on the resources that are protected and the role of the entity authorized for access. Primary considerations in establishing the cryptoperiod for private authorization keys include the robustness of the key, the adequacy of the cryptographic method, and the adequacy of key protection mechanisms and procedures. The cryptoperiod of the private authorization key and its associated public key **shall** be the same.

b. Cryptoperiod: Given the use of FIPS-approved algorithms and key sizes, and an expectation that the security of the key storage and use environment will increase as the sensitivity and criticality of the authorization processes increases, it is recommended that cryptoperiods for private authorization keys be less than two years.

19. *Public authorization key*:

a. Type Considerations: A public authorization key is the public element of an asymmetric key pair used to verify privileges for an entity that possesses the associated private key. The length of the public authorization key cryptoperiod is of far less concern, from a security point of view, than is that of the associated private key.

b. Cryptoperiod: The cryptoperiod of the public authorization key **shall** be the same as the authorization private key: less than two years.

8.2.6 Recommendations for Other Keying Material

Other keying material does not have well-established cryptoperiods, per se. The following guidelines are offered regarding the disposition of this other keying material:

1. Domain parameters remain in effect until changed.
2. An IV is associated with the information that it helps to protect, and is needed until the information and its protection are no longer needed.
3. Shared secrets **shall** be destroyed when no longer needed to derive keying material.
4. Secret seeds **should** be destroyed immediately after use.
5. Public seeds **should not** be retained longer than needed for validation (e.g., as one of the elliptic curve domain parameters).
6. Other public information **should not** be retained longer than needed for cryptographic processing.
7. Intermediate results **shall** be destroyed immediately after use.

8.3 Assurance of Domain Parameter and Public Key Validity

An assurance of the validity of the domain parameters and public keys are important to applications of public key cryptography and **shall** be obtained.

Domain parameters are used by some public key algorithms during the generation of key pairs, digital signatur es and shared secrets that are subsequently used to derive keying material. Invalid domain parameters could void all intended security for all entities using the domain parameters. Methods of obtaining assurance of domain parameter validity for DSA, and finite field discrete log and MQV key agreement algorithms are provided in [FIPS 186-3], [SP-800-56] and [ANSI X9.42]. Methods for obtaining this assurance for ECDSA, and the elliptic curve discrete log and MQV key establishment algorithms are provided in [sp 800-56], [ANSIX9.62], and [ANSI X9.63].

Invalid public keys could result in voiding the intended security, including the security of the operation (i.e., digital signature, key establishment, encryption), leaking some or all information from the owner's private key, and leaking some or all information about a private key that is combined with an invalid public key (as may be done when key agreement or public key encryption is performed). Methods of obtaining assurance of public validity for DSA, and finite field discrete log and MQV key agreement algorithms are provided in [FIPS 186-3], [SP 800-56] and [ANSI X9.42]. Methods for obtaining this assurance for ECDSA, and the elliptic curve discrete log and MQV key establishment algorithms are provided in [SP 800-56], [ANSIX9.62], and [ANSI X9.63].

8.4 Compromise of Keys and other Keying Material

Information protected by cryptographic mechanisms is secure only if the algorithms remain strong, and the keys have not been compromised. Key compromise occurs when the protective mechanisms for the key fail (e.g., the confidentiality, integrity or association of the key to its owner fail - see Sections 5), and the key can no longer be trusted to provide the required security. When a key is compromised, all use of the key to apply the protection mechanism on information (e.g., compute a digital signature or encrypt information) **shall** cease, the compromised key **shall** be revoked, and all entities using or relying on that key **shall** be notified (see Section 7.3.5). However, continued use of the key to remove or verify the protections (e.g., decrypt or verify a digital signature) may be warranted, depending on the risks of continued use and an organization's Key Management Policy (see Part 2). Limiting the cryptoperiod of the key limits the amount of material that would be compromised (exposed) if the key were compromised. Using different keys for different purposes (e.g., different applications as well as different cryptographic mechanisms) as well as limiting the amount of information protected by a single key also achieves this purpose.

The compromise of a key has the following implications:

1. A compromise of the confidentiality of a key means that another entity (an unauthorized entity) may know the key and be able to use that key to perform computations requiring the use of the key. In general, the compromise of a key used to provide confidentiality

protection[12] (i.e., via encryption) means that all information encrypted by that key could be known by unauthorized entities. In addition, the encrypted information could contain false information that was originated by an unauthorized entity (an entity that is not authorized to know the key).

For example, if a symmetric data encryption key is compromised, the unauthorized entity might use the key to decrypt past or future encrypted information, i.e., the information is no longer confidential between the authorized entities. The unauthorized entity might masquerade as a legitimate entity and send false information, i.e., the authenticity and source of the information would be in question.

As another example, if a private signature key is compromised, the unauthorized entity might sign messages as if they were originated by the key's real owner (either new messages or messages that are altered from their original contents), i.e., non-repudiation and authenticity of the information is in question.

2. A compromise of the integrity of a key means that the key is incorrect - either that the key has been modified (either deliberately or accidentally), or that another key has been substituted; this includes a deletion (non-availability) of the key. The compromise of a key used to provide integrity[13] calls into question the integrity of all information protected by the key. This information could have been provided by, or changed by, an unauthorized entity.
3. A compromise of a key's usage or application association means that the key could be used for the wrong purpose (e.g., key establishment instead of digital signatures) or for the wrong application, and could result in the compromise of information protected by the key.
4. A compromise of a key's association with the owner or other entity means that the identity of the other entity cannot be assured (i.e., one doesn't know who the other entity really is) or that information cannot be processed correctly (e.g., encrypted or decrypted with the correct key).
5. A compromise of a key's association with other information means that there is no association at all, or the association is with the wrong "information". This could cause the cryptographic services to fail, information to be lost, or the security of the information to be compromised.

Certain protective measures may be taken in order to minimize the likelihood or consequences of a key compromise. The following procedures are usually involved:

a. Limiting the amount of time a symmetric or private key is in plaintext form.

[12] As opposed to the confidentiality of a key that could, for example, be used as a signing private key.

[13] As opposed to the integrity of a key that could, for example, be used for encryption.

b. Restricting plaintext symmetric and private keys to approved key “containers”. This includes key generators, key transport devices, key loaders, cryptographic modules, and key storage devices.

c. Preventing humans from viewing plaintext symmetric and private keys.

d. Using integrity checks to assure that the integrity of a key or its association with other data has not been compromised. For example, keys may be wrapped (i.e., encrypted) in such a manner that unauthorized modifications to the wrapping or to the associations will be detected.

e. Employing key confirmation and receipt to help ensure that the proper key was, in fact, established.

f. Establishing an accountability system that keeps track of each access to symmetric and private keys in plaintext form.

g. Providing a cryptographic integrity check on the key (e.g., a MAC or a digital signature).

h. The use of trusted time stamps for signed data.

i. Destroying keys as soon as they are no longer needed.

The worst form of key compromise is one that is not detected. Nevertheless, even in this case, certain protective measures can be taken. Key management systems (KMS) **should** be designed to mitigate the negative effects of a key compromise. A KMS **should** be designed so that the compromise of a single key compromises as few other keys as possible. For example, a single cryptographic key could be used to protect the data of only a single user or a limited number of users, rather than a large number of users. Often, systems have alternative methods to authenticate communicating entities that do not rely solely on the possession of keys. The object is to avoid building a system with catastrophic weaknesses.

A compromise recovery plan (see Section 8.7.4) is essential for restoring cryptographic security services in the event of a key compromise. A compromise recovery plan **shall** be documented and easily accessible. The plan may be included in the Key Management Practices Statement (see Part 2). If not, the Key Management Practices Statement **should** reference the compromise recovery plan.

Although compromise recovery is primarily a local action, the repercussions of a compromised key are shared by the entire community that uses the system or equipment. Therefore, compromise recovery procedures **should** include the community at large. For example, recovery from the compromise of a root CA’s private signature key requires that all users of the infrastructure obtain and install a new trust anchor. Typically, this involves physical procedures that are expensive to implement. To avoid these expensive procedures, elaborate precautions to avoid compromise may be justified.

The compromise recovery plan **should** contain:

1. The identification of the personnel to notify,
2. The identification of the personnel to perform the recovery actions,
3. The rekey or key replacement method, and
4. Any other recovery procedures.

Other compromise recovery procedures may include:

5. Physical inspection of the equipment,
6. Identification of all information that may be compromised as a result of the incident,
7. Identification of all signatures that may be invalid due to the compromise of a signing key, and
8. Distribution of new keying material, if required.

8.5 Accountability

Accountability involves the identification of those entities who have access to, or control of, cryptographic keys throughout their lifecycles. Accountability can be an effective tool to help prevent key compromises and to reduce the impact of compromises once they are detected. As a minimum, the key management system **should** account for all individuals who are able to view plaintext cryptographic keys. In addition, more sophisticated key management systems may account for all individuals authorized access to, or control of, any cryptographic keys, whether in plaintext or ciphertext form. For example, a sophisticated accountability system might be able to determine each individual who had control of any given key over its entire life span. This would include the person in charge of generating the key, the person who used the key to cryptographically protect data, and the person who was responsible for destroying the key when it was no longer needed. Even though these individuals never actually saw the key in plaintext form, they are held accountable for the actions that they performed on or with the key.

Accountability provides three significant advantages:

1. It aids in the determination of when the compromise could have occurred and what individuals could have been involved,
2. It tends to protect against compromise because individuals with access to the key know that their access to the key is known, and
3. It is very useful in recovering from the detected key compromise to know where the key was used and what data, or other keys, were protected by the compromised key.

Certain principles have been found to be useful in enforcing the accountability of cryptographic keys. These principles might not apply to all systems or all types of keys. Some of the principles apply to longer-term keys that are controlled by humans. The principles include:

a. Uniquely identifying keys,
b. Identifying the key user,
c. Identifying dates and times of key use along with the data that is protected, and
d. Identifying other keys that are protected by a symmetric or private key.

8.6 Audit

Two types of audit **should** be performed on key management systems:

1. The security plan and the procedures that are developed to support the plan **should** be periodically audited to ensure that they continue to support the Key Management Policy (see Part 2).
2. The protective mechanisms employed **should** be periodically reassessed with respect to the level of security that they provide and are expected to provide in the future, and that the mechanisms correctly and effectively support the appropriate policies. New technology developments and attacks **should** be taken into consideration.

On a more frequent basis, the actions of the humans that use, operate and maintain the system **should** be reviewed to verify that the humans continue to follow established security procedures. Strong cryptographic systems can be compromised by lax and inappropriate human actions.

8.7 Key Management System Survivability

8.7.1 Back-up Keys

[OMB 11/01] notes that encryption is an important tool for protecting the confidentiality of disclosure-sensitive information that is entrusted to an agency's care, but that the encryption of agency data also presents risks to the availability of information needed for mission performance. Agencies are reminded of the need to protect the continuity of their information technology operations and agency services when implementing encryption. The guidance specifically notes that, without access to the cryptographic keys that are needed to decrypt information, organizations risk the loss of their access to that information. Consequently, it is prudent to retain back-up copies of the keys necessary to decrypt stored enciphered information, including master keys, key encrypting keys, and the related keying material necessary to decrypt encrypted information until there is no longer any requirement for access to the underlying plaintext information (see Tables 4-5).

As the tables show, there are other keys for the operations of some organizations that may require the retention of back-up copies (e.g. public signature verification keys and authorization keys). Back-up copies of keying material **shall** be stored in accordance with the provisions of Section 5 in order to protect the confidentiality of encrypted information and the integrity of source authentication, data integrity, and authorization processes.

8.7.2 Key Recovery

There are a number of issues associated with key recovery. An extensive discussion is provided in Appendix B. Key recovery issues to be addressed include:

1. Which keying material, if any, needs to be backed up or archived for later recovery?
2. Where will backed up or archived keying material be stored?
3. Who will be responsible for protecting the backed up or archived keying material?
4. What procedures need to be put in place for storing and recovering the keying material?
5. Who can request a recovery of the keying material and under what conditions?
6. Who will be notified when a key recovery has taken place and under what conditions?
7. What audit or accounting functions need to be performed to ensure that the keying material is only provided to authorized entities?

8.7.3 System Redundancy/Contingency Planning

Cryptography is a useful tool for preventing unauthorized access to data and/or resources, but when the mechanism fails, it can prevent access by valid users to critical information and processes. Loss or corruption of the only copy of cryptographic keys can deny users access to information. For example, a locksmith can usually defeat a broken physical mechanism, but access to information protected by a strong algorithm may not be practical without the correct decryption key. Continuity of an organization's operations can depend heavily on contingency planning for key management systems that includes a redundancy of critical logical processes and elements, including key management and cryptographic keys.

8.7.3.1 General Principles

Planning for recovery from system failures is an essential management function. Interruptions of critical infrastructure services **should** be anticipated, and planning for maintaining the continuity of operations in support of an organization's primary mission requirements **should** be accomplished. With respect to key management, the following situations are typical of those for which planning is necessary:

1. Lost key cards or tokens,
2. Forgotten passwords that control access to keys,
3. Failure of key input devices (e.g., readers),
4. Loss or corruption of the memory media on which keys and/or certificates are stored,
5. Compromise of keys,
6. Corruption of Certificate Revocation Lists (CRL) or Compromised Key Lists (CKLs),
7. Hardware failure of key or certificate generation, registration, and/or distribution systems, subsystems, or components,
8. Power loss requiring re-initialization of key or certificate generation, registration, and/or distribution systems, subsystems, or components,
9. Corruption of the memory media necessary to key or certificate generation, registration, and/or distribution systems, subsystems, or components,
10. Corruption or loss of key or certificate distribution records and/or audit logs, and
11. Loss or corruption of association of keying material to the holders/users of the keying material.
12. Unavailability of older software or hardware that are needed to access keying material or process protected information.

While recovery discussions most commonly focus on recovery of encrypted data and the restoration of encrypted communications capabilities, planning **should** also address 1) the restoration of access (without creating temporary loss of access protections) where cryptography is used in access control mechanisms, 2) the restoration of critical processes (without creating temporary loss of privilege restrictions) where cryptography is used in authorization mechanisms, and 3) the maintenance/restoration of integrity protection in digital signature and message authentication applications.

Contingency planning **should** include 1) providing a means and assigning responsibilities for rapidly recognizing and reporting critical failures; 2) the assignment of responsibilities and placement of resources for bypassing or replacing failed systems, subsystems, and components; and 3) the establishment of detailed bypass and/or recovery procedures.

Contingency planning includes a full range of integrated logistics support functions. Spare parts (including copies of critical software programs, manuals, and data files) **should** be available (acquired or arranged for) and pre-positioned (or delivery staged). Emergency maintenance, replacement, and/or bypass instructions **should** be prepared and disseminated to both designated individuals and to an accessible and advertised access point. Designated individuals **should** be trained in their assigned recovery procedures, and all personnel **should** be trained in reporting procedures and workstation-specific recovery procedures.

8.7.3.2 Cryptography and Key Management-specific Recovery Issues

Cryptographic keys are relatively small components or data elements that often control access to large volumes of information or critical processes. As OMB has noted [OMB11/01], "without access to the cryptographic key(s) needed to decrypt information [an] agency risks losing access to its valuable information." Agencies are reminded of the need to protect the continuity of their information technology operations and agency services when implementing encryption. The guidance particularly stresses that agencies must address information availability and assurance requirements through appropriate data recovery mechanisms such as cryptographic key recovery.

Key recovery generally involves some redundancy, or multiple copies of keying material. If one copy of a critical key is lost or corrupted, another copy usually needs to be available in order to recover data and/or restore capabilities. At the same time, the more copies of a key that exist and are distributed to different locations, the more susceptible the key usually is to compromise through penetration of the storage location or subversion of the custodian (e.g., user, service agent, key production/distribution facility). In this sense, key confidentiality requirements conflict with continuity of operations requirements. Special care needs to be taken to safeguard all copies of keying material, especially symmetric keys and private asymmetric keys. Where privacy is a concern, this includes strict control of access to duplicate material (e.g., split knowledge).

More detail regarding contingency plans and planning requirements is provided in Part 2 of this *Key Management Guideline*.

8.7.4 Compromise Recovery

When keying material that is used to protect sensitive information or critical processes is disclosed to unauthorized entities, all of the information and/or processes protected by that keying material becomes immediately subject to disclosure, modification, subversion, and/or denial of service. All affected parties **shall** be notified immediately; all affected keys **shall** be replaced; and, where sensitive or critical information or processes are affected, an immediate damage assessment **should** be conducted. Measures necessary to mitigate the consequences of suspected unauthorized access to protected data or processes and to reduce the probability or frequency of future compromises may follow.

Where symmetric keys or private asymmetric keys are used to access only a single user's information or communications between a single pair of users, the compromise recovery process can be relatively simple and inexpensive. Relatively few individuals need to be notified, and relatively few keys need to be changed. Damage assessment and mitigation measures are often local matters.

On the other hand, where a key is shared by or affects a large number of users, damage can be widespread, and recovery is both complex and expensive. Some examples of keys, the compromise of which might be particularly difficult or expensive to recover from, include the following:

1. Private key used to sign a root certificate in a public key infrastructure
2. Symmetric key transport key shared by a large number of users
3. Private asymmetric key transport key shared by a large number of users
4. Master key used in the generation of keys by a large number of users
5. Symmetric data encryption key used to encrypt data in a large distributed database
6. Symmetric key shared by a large number of communications network participants
7. Key used to protect a large number of stored keys
8. A certification authority's (CA's) private key

In all of these cases, a large number of holders would need to be immediately notified of the compromise. Inclusion of the key identity on a Compromised Key List (CKL) or Certificate Revocation List (CRL) to be published at a later date would not be sufficient. This means that a list of holders would need to be maintained and a means for communicating news of the compromise of the holders would be required. Unless the communications path is secure, news of the compromise might be available to a wide audience and might tempt many in that audience to attempt to exploit the compromise.

In all of these cases, a secure path for replacing the compromised keys is be required. In order to permit rapid restoration of service, an electronic (e.g., over-the-air) replacement path is preferred (see Section 7.2.3). In some cases, however, there may be no practical alternative to manual distribution (e.g., compromise of a root CA's private key). Contingency distribution of alternate keys may help restore service rapidly in some circumstances (e.g., compromise of a widely held symmetric key), but the possibility of simultaneous compromise of operational and contingency keys would need to be considered.

Damage assessment can be extraordinarily complex, particularly in cases such as CA private keys, widely used transport keys, and keys used by many users of large distributed databases.

8.8 Guidance for Cryptographic Algorithm and Key Size Selection

Cryptographic algorithms that provide the security services identified in Section 3 are specified in Federal Information Processing Standards (FIPS) and NIST Recommendations. Several of these algorithms are defined for a number of key sizes. This section provides guidance for the selection of appropriate algorithms and key sizes.

This section emphasizes the importance of acquiring cryptographic systems with appropriate algorithm and key sizes to provide adequate protection for 1) the expected lifetime of the system and 2) any data protected by that system during the expected lifetime of the data.

8.8.1 Equivalent Algorithm Strengths

Cryptographic algorithms provide different "strengths" of security, depending on the algorithm and the key size used. In this discussion, two algorithms are considered to be of equivalent strength for the given key sizes (*X* and *Y*) if the amount of work needed to "break the algorithms" or determine the keys (with the given key sizes) is approximately the same using a given resource. The strength of an algorithm (sometimes called the work factor) for a given key size is traditionally described in terms of the amount of work it takes to try all keys for a symmetric algorithm with a key size of "*X*" that has no short cut attacks (i.e., the most efficient attack is to try all possible keys). In this case, the best attack is said to be the exhaustion attack. An algorithm that has a "*Y*" bit key, but whose strength is equivalent to an "*X*" bit key of such a symmetric algorithm is said to provide "*X* bits of security" or to provide "*X*-bits of strength". An algorithm that provides *X* bits of strength would, on average, take $2^{X-1}T$ to attack, where *T* is the amount of time that is required to perform one encryption of a plaintext value and comparison of the result against the corresponding ciphertext value.

Determining the security strength of an algorithm can be nontrivial. For example, consider TDES. TDES uses three 56-bit keys (*K*1, *K*2 and *K*3). If each of these keys is independently generated, then this is called the three key option or three key TDES (3TDES). However, if *K*1 and *K*2 are independently generated, and *K*3 is set equal to K1, then this is called the two key option or two key TDES (2TDES). One might expect that 3TDES would provide $56 \times 3 = 168$ bits of strength. However, there is an attack on 3TDES that reduces the strength to the work that would be involved in exhausting a 112-bit key. For 2TDES, if exhaustion were the best attack, then the strength of 2TDES would be $56 \times 2 = 112$ bits. This appears to be the case if the attacker has only a few matched plain and cipher pairs. However, if the attacker can obtain approximately 2^{40} such pairs, then 2TDES has strength equivalent to an 80-bit algorithm (see [ANSIX9.52], Annex B).

The recommended key size equivalencies discussed in this section are based on assessments made as of the publication of this guideline. Advances in factoring algorithms, advances in general discrete logarithm attacks, elliptic curve discrete logarithm attacks and quantum computing may affect these equivalencies in the future. New or improved attacks or technologies may be developed that leave some of the current algorithms completely insecure. In the case of quantum computing, the asymmetric techniques may no longer be secure. Periodic reviews will be performed to determine whether the stated equivalencies need to be revised (e.g., the key sizes need to be increased) or the algorithms are no longer secure.

When selecting a block cipher cryptographic algorithm (e.g., AES or TDES), the block size may also be a factor that should be considered, since the amount of security provided by several of the modes defined in [SP 800-38] is dependent on the block size[14]. More information on this issue is provided in [SP 800-38].

Table 8 provides equivalence guidelines for the Approved algorithms.

---

[14] Suppose that the block size is *b* bits. The collision resistance of a MAC is limited by the size of the tag and collisions become probable after $2^{b/2}$ messages, if the full *b* bits are used as a tag. When using the Output Feedback mode of encryption, the maximum cycle length of the cipher can be at most 2^b blocks; the average cipher length is less than 2^b blocks. When using the Cipher Block Chaining mode, plaintext information is likely to begin to leak after $2^{b/2}$ blocks have been encrypted with the same key.

1. Column 1 indicates the number of bits of security provided by the algorithms and key sizes in a particular row. Note that the “bits of security” is not necessarily the same as the key sizes for the algorithms in the other columns, due to attacks on those algorithm that provide computational advantages.
2. Column 2 identifies the symmetric key algorithms that provide the indicated level of security (at a minimum), where 2TDES and 3TDES are approved in [FIPS 46-3] and specified in [ANSI X9.52], and AES is specified in [FIPS 197]. 2TDES is TDES with two different keys; 3TDES is TDES with three different keys.
3. Column 3 provides the equivalent hash functions that are specified in FIPS180-2 for the given level of security for applications where “collisions” are a concern (e.g., for digital signature and MAC applications). For these applications, the maximum strength is achieved when at least *b*/2 bits of random data are input into the hash function, where *b* is the size of the output block of the hash function. [Are there any other applications for which collisions are a concern?]
4. Column 4 provides the equivalent hash functions that are specified in FIPS180-2 for the given level of security for applications where “collisions” are not a concern (e.g., for the computation of random numbers in some cases). For these applications, the maximum strength is achieved when at least *b* bits of random data are input into the hash function, where *b* is the size of the output block of the hash function. [Are there other applications that do not have collision concerns?]
5. Column 5 indicates the size of the parameters associated with the standards that use discrete logs and finite field arithmetic (DSA as defined in [FIPS186-3] for digital signatures, and Diffie-Hellman (DH) and MQV key agreement as defined in [ANSI X9.42] and [SP 800-56]), where *L* is the size of the modulus *p* and the public key, and *N* is the size of *q* and the private key.
6. Column 6 defines the value for *k* (the size of the modulus *n*) for the RSA algorithm specified in [ANSI X9.31] and [PKCS #1] and adopted in [FIPS 186-2] for digital signatures, and specified in [ANSI X9.44] and adopted in [SP 800-56] for key establishment. The value of *k* is commonly considered to be the key size.
7. Column 7 defines the order of *f* (the size of *n*, where *n* is the order of the base point *G*) for the discrete log algorithms using elliptic curve arithmetic that are specified for digital signatures in [ANSI X9.62] and adopted in [FIPS 186-2], and for key establishment as specified in [ANSIX9.63] and adopted in [SP 800-56]. The value of *f* is commonly considered to be the key size.

Table 8: Equivalent strengths[15].

Bits of security	Symmetric key algs.	Hash functions (collision concerns)	Hash functions (no collision concerns)	DSA, D-H, MQV	RSA	Elliptic Curves
80	2TDES[16]	SHA-1		$L = 1024$ $N = 160$	$k = 1024$	$f = 160$
112	3TDES			$L = 2048$ $N = 224$	$k = 2048$	$f = 224$
128	AES-128	SHA-256		$L = 3072$ $N = 256$	$k = 3072$	$f = 256$
160			SHA-1			
192	AES-192	SHA-384		$L = 7680$ $N = 384$	$k = 7680$	$f = 384$
256	AES-256	SHA-512	SHA-256	$L = 15360$ $N = 512$	$k = 15360$	$f = 512$
384			SHA-384			
512			SHA-512			

8.8.2 Defining Appropriate Algorithm Suites

Many applications require the use of several different cryptographic algorithms. When several algorithms can be used to perform the same service, some algorithms are inherently more efficient because of their design (e.g., AES has been designed to be more efficient than Triple DES).

In many cases, a variety of key sizes may be available for an algorithm. For some of the algorithms (e.g., public key algorithms, such as RSA), the use of larger key sizes than are required may impact operations, e.g., larger keys may take longer to generate or longer to process the data. However, the use of key sizes that are too small may not provide adequate security.

Table 9 provides recommendations that may be used to select an appropriate suite of algorithms and key sizes for Federal Government unclassified applications. A minimum of eighty bits of security **shall** be provided by most applications until 2015. Between 2016 and 2035, a minimum

[15] When no algorithm or key size is available to provide a given security strength, the cell is left empty.

[16] Depending upon the availability of matched plaintext and cipher text, the strength of 2TDES can range from 80 to 112 bits.

of 112 bits of security **shall** be provided. Thereafter, that at least 128 bits of security **shall** be provided.

1. Column 1 indicates the years during which the algorithms specified in subsequent columns are appropriate for use.
2. Column 2 identifies appropriate symmetric key algorithms and key sizes: 2TDES and 3TDES are specified in [FIPS46-3], the AES algorithm is specified in [FIPS 197], and the computation of Message Authentication Codes (MACs) using block ciphers is specified in [SP 800-38B].
3. Column 3 specifies the hash functions to be used for applications where "collisions" are a concern (e.g., digital signatures and HMAC). Hash functions are specified in [FIPS 180-2]. [List the applications?]
4. Column 4 specifies the hash functions to be used for applications where "collisions" are not a concern (e.g., random numbers). Hash functions are specified in [FIPS 180-2]. This column assumes that at least *s* bits of randomness are input into the hash function, where *s* is the minimum security strength for the time period. [List the applications?]
5. Column 5 indicates the minimum size of the parameters associated with DSA as defined in FIPS [186-3].
6. Column 6 defines the minimum size of the modulus for the RSA algorithm specified in [ANSI X9.31] and [PKCS #1] and adopted in [FIPS 186-2] for digital signatures, and specified in ANSI X9.44 and adopted in [SP 800-56] for key establishment.
7. Column 7 defines the minimum size of the base point for the elliptic curve algorithms specified for digital signatures in [ANSI X9.62] and adopted in [FIPS 186-2], and for key establishment as specified in [ANSI X9.63] and adopted in [SP 800-56].

Table 9: Recommended algorithms and minimum key sizes.

Years	**Symmetric key algs. (Encryption & MAC)**	**Hash functions (collisions)**	**Hash functions (no collisions)**	**DSA, D-H, MQV**	**RSA**	**Elliptic Curves**
Present - 2015 (min. of 80 bits of strength)	2TDES 3TDES AES-128 AES-192 AES-256	SHA-1 SHA-256 SHA-384 SHA-512	SHA-1 SHA-256 SHA-384 SHA-512	Min.: L = 1024; N =160	Min.: k=1024	Min.: f=160
2016 - 2035 (min. of 112 bits of strength)	3TDES AES-128 AES-192 AES-256	SHA-256 SHA-384 SHA-512	SHA-1 SHA-256 SHA-384 SHA-512	Min.: L = 2048 N = 224	Min.: k=2048	Min.: f=224

Years	**Symmetric key algs. (Encryption & MAC)**	**Hash functions (collisions)**	**Hash functions (no collisions)**	**DSA, D-H, MQV**	**RSA**	**Elliptic Curves**
2036 and beyond (min. of 128 bits of strength)	AES-128 AES-192 AES-256	SHA-256 SHA-384 SHA-512	SHA-1 SHA-256 SHA-384 SHA-512	Min.: $L = 3072$ $N = 256$	Min.: k=3072	Min.: f=256

[For 112 bit of security, should the output of SHA-256 be truncated to 224 bits, should truncation not be allowed, or should truncation be optional?]

The algorithms and key sizes in the table are considered appropriate for the protection of data during the given time periods. Algorithms or key sizes not indicated for a given range of years **shall not** be used to protect information during that time period. If the security life of information extends beyond one time period specified in the table into the next time period (the later time period), the algorithms and key sizes specified for the later time **shall** be used. The following examples are provided to clarify the use of the table:

a. If information is originally protected in 2005, and the maximum expected security life of that data is for only ten years, any of the algorithms or key sizes in the table may be used.

b. If a CA signature key and all certificates issued under that key will expire in five years, then the signature and hash algorithm used to sign the certificate needs to be secure for at least five years. A certificate issued in 2005 using 1024 bit DSA and SHA-1 would be acceptable; issuing a certificate in 2015 using the same algorithm, key size and hash function would not provide adequate security for the certificate.

c. If information is initially signed in 2014 and needs to remain secure for a maximum of ten years (i.e., from 2014 to 2023), SHA-1 or a 1024 bit RSA key would not provide sufficient protection between 2016 and 2023 and, therefore, it is not recommended that SHA-1 or 1024-bit RSA be used. It is recommended that the algorithms and key sizes in the "2016-2035" row should be used to provide the cryptographic protection.

d. If information is cryptographically protected in 2013, and needs to remain secure for 30 years (i.e., from 2013 to 2042), then it is recommended that the algorithms and key sizes in the "2036 and Beyond" row be used.

8.8.3 Using Algorithm Suites

Algorithm suites that combine non-equivalent strength algorithms are generally discouraged. However, algorithms of different strengths and key sizes may be used together for performance, availability or interoperability reasons, provided that sufficient protection is provided. In general, the weakest algorithm and key size used to provide cryptographic protection determines the strength of the protection. Exceptions to this "rule" require extensive analysis. Determination of the security strength provided for protected information includes an analysis not only of the algorithm(s) and key size(s) used to apply the cryptographic protection(s) to the information, but also any algorithms and key sizes associated with establishing the key(s) used for information protection, including those used by communication protocols.

The following is a list of several algorithm combinations and discussions on the security implications of the combination:

1. When a key establishment scheme is used to establish keying material for use with one or more algorithms (e.g., TDES, AES, MAC, HMAC), the strength of the selected combination is equivalent to the weakest algorithm and key size used. For example, if a 1024 bit RSA key is used to establish a 128-bit AES key (as defined in [SP 800-56]), only 80 bits of security are provided for any information protected by that AES key, since the 1024 bit RSA provides only 80 bits of security. If 128 bits of security are required for the information protected by AES, then either an RSA key size of at least 3072 bits, or another key establishment algorithm and appropriate key size needs to be selected to provide the required protection.
2. When a hash function and digital signature algorithm are used in combination to compute a digital signature, the strength of the signature is determined by the weaker of the two algorithms. For example, SHA-256 used with ECDSA using a 160-bit key provides 80 bits of security because a 160-bit ECDSA key provides only 80 bits of security. If 112 bits of security is required, a 224-bit ECDSA key would be appropriate; if 128 bits of security is required, a 256-bit ECDSA key would be appropriate.
3. When encryption is used with a MAC or a digital signature to protect information, the protection provided for the information is equivalent to the protection provided by the weaker algorithm. For example, 3TDES used with a SHA-1 HMAC using an 80-bit key to protect a message, provides 80 bits of security; 3TDES used with a SHA-256 HMAC using a 128 bit key provides 112 bits of security; 3TDES used with a 3TDES MAC provides 112 bits of security. As another example, AES used with a 1024-bit DSA key provides 80 bits of security; AES used with a 2048-bit DSA key provides 112 bits of security.
4. [Other combinations to be considered?]

Typically, an organization selects the cryptographic services that are needed for a particular application. Then, based on the security life of the data and the years the system is anticipated to be in use, an algorithm and key size suite is selected that is sufficient to meet these requirements. The organization then establishes a key management system, including approved cryptographic products that provide the services required by the application. Finally, when the current algorithm and key size suite nears its expiration date as determined by the security life of the information to be protected (see Table 9), the organization transitions to a new algorithm and key size suite offering appropriate security.

If it is determined that a certain level of security is required for the protection of certain information, then an algorithm and key size suite needs to be selected that would provide that level of security as a minimum. For example, if 128 bits of security are required for information that is to be communicated and provided with confidentiality, integrity, authentication and non-repudiation protection, the following selection of algorithms and key sizes may be appropriate:

a. Confidentiality: Encrypt the information using AES-128. Other AES key sizes would also be appropriate, but perform a bit slower.

b. Integrity, authentication and non-repudiation: Suppose that only one cryptographic operation is preferred. Use digital signatures. SHA-256 could be selected for the hash function, although SHA-384 and SHA-512 are also appropriate, but the performance is slower. Select an algorithm for digital signatures from what is available to an application (e.g., DSA or RSA with at least a 3072-bit key, or ECDSA[17] with at least a 256-bit key). If more than one algorithm and key size is available, the selection may be based on algorithm performance, memory requirements, etc. as long as the minimum requirements are met.
c. Key establishment: Select a key establishment scheme based on the application and environment (see [SP 800-56]), the availability of an algorithm in an implementation, and its performance. Select a key size from Tables 8 and 9 for the algorithm that provides at least 128 bits of security. For example, if RSA is available, use an RSA key transport (i.e., key establishment) scheme with a 3072-bit key. However, it is recommended that the key used for key transport be different from an RSA key used for digital signatures.

Agencies that procure systems **should** consider the potential operational lifetime of the system. The agencies **shall** either select algorithms that are expected to be secure during the entire system lifetime, or **shall** ensure that the algorithms and key sizes can be readily updated.

8.8.4 Transitioning to New Algorithms and Key Sizes

There are many legacy applications currently available that use algorithms and key sizes not specified in Table 8. When the algorithm or key size is determined to no longer provide the desired protection for information (e.g., the algorithm has been "broken"), any information "protected" by the algorithm or key size is considered to be "exposed" (e.g., no longer confidential, or the integrity cannot be assured). It **should** be assumed that encrypted information could have been collected and retained by unauthorized entities (adversaries). At some time, the unauthorized entity may attempt to decrypt the information. Even when an algorithm and key size suite used by a system is replaced (e.g., by a different algorithm or key size), the information protected by the previous algorithm and key size suite is vulnerable. Thus, when using Table 9 to select the appropriate key size for an encryption algorithm, it is very important to take the expected security life of the data into consideration.

Transition from one algorithm or key size to another is difficult because the information that was considered to be protected by the previous algorithm or key size can no longer be considered as protected. Therefore, when initiating cryptographic protections for information, the strongest algorithm and key size that is appropriate for providing the protection **should** be used in order to delay transitions. However, it should be noted that selecting some algorithms, or key sizes that are unnecessarily large may have adverse affects (e.g., performance may be unacceptably slow).

8.9 Key Management Specifications for Cryptographic Devices or Applications

Key management is often an afterthought in the cryptographic development process. As a result, cryptographic subsystems too often fail to support the key management functionality and protocols that are necessary to provide adequate security with the minimum necessary reduction

[17] Currently, ECDSA only supports the use of SHA-1.

in operational efficiency. All cryptographic development activities **should** involve key management planning and specification (see Part 2). Key management planning **should** begin during the initial conceptual/development stages of the cryptographic development lifecycle, or during the initial discussion stages for cryptographic applications of existing cryptographic components into information systems and networks. The specifications that result from the planning activities **shall** be consistent with NIST key management guidance.

For cryptographic development efforts, a key specification and acquisition planning process **should** begin as soon as the candidate algorithm(s) and, if appropriate, keying material media and format have been identified. Key management considerations may affect algorithm choice, due to operational efficiency considerations for anticipated applications. For the application of existing cryptographic products for which no key management specification exists, the planning and specification processes should begin during device and source selection and continue through acquisition and installation.

The types of key management components that are required for a specific cryptographic device and/or for suites of devices used by organizations **should** be standardized to the maximum possible extent, and new cryptographic device development efforts **shall** comply with NIST key management guidelines. Accordingly, NIST criteria for the security, accuracy, and utility of key management components in electronic and physical forms **shall** be met. Where the criteria for security, accuracy, and utility can be satisfied with standard key management components (e.g., PKI), the use of those compliant components is encouraged. A developer may choose to employ non-compliant key management as a result of security, accuracy, utility, or cost considerations. However, such developments **should** conform as closely as possible to established key management guidelines.

8.9.1 Key Management Specification Description/Purpose

The Key Management Specification is the document that describes the key management components that may be required to operate a cryptographic device throughout its lifetime. Where applicable, the Key Management Specification also describes key management components that are provided by a cryptographic device. The Key Management Specification documents the capabilities that the cryptographic application requires from key sources (e.g., the Key Management Infrastructure (KMI) described in Part 2 of this *Key Management Guideline*.

8.9.2 Content of the Key Management Specification

The level of detail required for each section of the Key Management Specification can be tailored, depending upon the complexity of the device or application for which the Key Management Specification is being written. The Key Management Specification **should** contain a title page that includes the device identity, and the developer's or integrator's identity. A revision page, list of reference documents, table of contents, and definition of abbreviations and acronyms page **should** also normally be included. The terminology used in a Key Management Specification **shall** be in accordance with the terms defined in appropriate NIST standards and guidelines. The Key Management Specification **should not** contain proprietary information. [Note: If the cryptographic application is supported by a PKI, a statement to that effect **should** be included in the appropriate Key Management Specification section below.]

8.9.2.1 Cryptographic Application and Communications Environment Description

Cryptographic Application and the Communications Environment Descriptions may provide a basis for the development of the rest of the Key Management Specification. The Cryptographic Application Description section provides a brief description of the cryptographic application or proposed employment of the cryptographic device. This includes the purpose or use of the cryptographic device (or application of a cryptographic device), and whether it is a new cryptographic device, a modification of an existing cryptographic device, or an existing cryptographic device for which a Key Management Specification does not exist. A brief description of the security services (confidentiality, integrity, non-repudiation, access control, identification and authentication, and availability) that the cryptographic device/application provides **should** be included. Information concerning long-term and potential interim key management support (key management components) for the cryptographic application **should** be provided.

8.9.2.2 Communications Environment

The Communications Environment section provides a brief description of the communications environment in which the cryptographic device is designed to operate. Some examples of communications environments include:

1. Data networks (intranet, internet, VPN),
2. Wired communications (landline, dedicated or shared switching resources), and
3. Wireless communications (satellite, radio frequency).

The environment may also include any anticipated access controls on communications resources, data sensitivity, privacy issues, non-repudiation requirements, etc. A figure that illustrates the communications environment and supporting text may be included in this section.

8.9.2.3 Key Management Component Requirements

The key management component requirements section describes the types and logical structure of keying material required for operation of the cryptographic device. Cryptographic applications using public key certificates (i.e., X.509 certificates) **should** describe the types of certificates supported. The following information **should** be included:

1. The different keying material classes or types required, supported, and/or generated;
2. The key management algorithm(s) (the applicable FIPS);
3. The keying material format(s) (reference any existing key specification if known);
4. The set of acceptable PKI policies (as applicable);
5. Tokens to be used.

The description of the key management component format may reference an existing cryptographic device key specification. If the format of the key management components is not already specified, then the format and medium **should** be specified in the Key Management Specification.

8.9.2.4 Key Management Component Generation

This section of the Key Management Specification **should** describe requirements for the generation of key management components by the cryptographic device for which the Key Management Specification is written. If the cryptographic device does not provide generation capabilities, identify the key management components that will be required from external sources **should** be identified.

8.9.2.5 Key Management Component Distribution

Where a device supports the automated distribution of keying material, this section of the Key Management Specification **should** describe the distribution and transport encapsulation (where employed) of keying material supported by the device. The distribution plan may describe the circumstances under which the key management components are encrypted or unencrypted, their physical form (electronic, PROM, floppy, disk, paper, etc.), and how they are identified during the distribution process. In the case of a dependence on manual distribution, the dependence and any handling assumptions regarding keying material **should** be stated.

8.9.2.6 Keying Material Storage

This section of the Key Management Specification **should** address how the cryptographic device or application for which the Key Management Specification is being written stores information, and how the keying material is identified during its storage life (e.g., Distinguished Name). The storage capacity capabilities for information **should** be included.

8.9.2.7 Access Control

This section of the Key Management Specification **should** address how access to the cryptographic device components and functions is to be authorized, controlled, and validated to request, generate, handle, distribute, store, and/or use keying material. Any use of passwords and personal identification numbers (PINs) **should** be included. For PKI cryptographic applications, role-based privileging and the use of tokens **should** be described.

8.9.2.8 Accounting

This section of the Key Management Specification **should** describe any device or application support for accounting for keying material. Any support for or outputs to logs used to support the tracking of key management component generation, distribution, storage, use and/or destruction **should** be detailed. The use of appropriate privileging to support the control of keying material that is used by the cryptographic application **should** also be described, in addition to the directory capabilities used to support PKI cryptographic applications, if applicable. The Key Management Specification **shall** identify where human and automated tracking actions are required and where two-person integrity is required, if applicable. Section 8.5 of this Recommendation provides accountability guidance.

8.9.2.9 Compromise Management and Recovery

This section of the Key Management Specification **should** address any support for the restoration of protected communications in the event of the compromise of keying material used by the cryptographic device/application. The recovery process description **should** include the methods key replacement. For PKI cryptographic applications, the implementation of Certificate Revocation Lists (CRLs) and Compromised Key Lists (CKLs) **should** be detailed. For system specifications, a description of how certificates will be reissued and renewed within the cryptographic application **should** also be included. General compromise recovery guidance is provided in Section 8.7.4 of this guideline.

8.9.2.10 Key Recovery

This section of the Key Management Specification describes product support or system mechanisms for effecting key recovery. Key recovery addresses how unavailable encryption keys can be recovered. System developers **should** include a discussion of the generation, storage, and access for long-term storage keys in the key recovery process description. The process of transitioning from the current to future long-term storage keys **should** also be included. General contingency planning guidance is provided in Section 8.7.3 of this guideline, System Redundancy/Contingency Planning. Key recovery is treated in detail in Appendix B, Key Recovery.

8.10 Key Establishment Schemes

Schemes for the establishment of keying material are provided in [SP 800-56]. Methods have been provided for both key agreement and key transport. Key agreement schemes use asymmetric (public key) techniques. Key transport schemes use either symmetric key or asymmetric key techniques. Key transport schemes using symmetric techniques are commonly known as key wrapping algorithms.

This section will contain material that is pertinent to [SP 800-56] that is not addressed elsewhere in this Recommendation. The following is a possible list of information to be included:

1. Discussions on using the schemes in SP 800-56.
2. Constraints associated with the use of the key wrapping algorithm, e.g., don't encrypt a 256 bit key with a 128 bit key.
3. Discuss padding
4. Provide guidance on the use of key confirmation
5. Provide a warning in response to Don Johnson's comment: Given all the security requirements stated in the schemes document, some mention of the potential for side-effect attacks seems warranted, as this can unravel a secret/private key by monitoring energy, time, radio waves, etc.

APPENDIX A: Cryptographic and Non-cryptographic Integrity and Authentication Mechanisms

Integrity and authentication services are particularly important in protocols that include key management. When integrity or authentication services are discussed in this guideline, they are afforded by “strong” cryptographic integrity or authentication mechanisms. Secure communications and key management are typically provided using a communications protocol that offers certain services, such as integrity or a "reliable" transport service. However, the integrity or reliable transport services of communications protocols are not necessarily adequate for cryptographic applications, particularly for key management, and there might be confusion about the meaning of terms such as “integrity”.

All communications channels have some noise (i.e., unintentional errors inserted by the transmission media), and other factors, such as network congestion, can cause packets to be lost. Therefore, integrity and reliable transport services for communications protocols are designed to function over a channel with certain worst-case noise characteristics. Transmission bit errors are typically detected using 1) a non-cryptographic checksum[18] to detect transmission errors in a packet, and 2) a packet counter that is used to detect lost packets. A receiving entity that detects damaged packets (i.e., packets in which errors have been detected) or lost packets may request the sender to retransmit them. The non-cryptographic checksums are generally effective at detecting random noise. For example, the common CRC-32 checksum algorithm used in LAN applications detects all error bursts with a span of less than 32 bits, and detects longer random bursts with a 2^{-32} failure probability. However, the non-cryptographic CRC-32 checksum does not detect the swapping of 32-bit message words, and specific errors in particular message bits cause predictable changes in the CRC-32 checksum. The sophisticated attacker can take advantage of this to create altered messages that pass the CRC-32 integrity checks, even, in some cases, when the message is encrypted.

Forward error correcting codes are a subset of non-cryptographic checksums that can be used to correct a limited number of errors without retransmission. These codes may be used, depending on the application and noise properties of the channel.

Cryptographic integrity mechanisms, on the other hand, protect against an active, intelligent attacker who might attempt to disguise his attack as noise. The bits altered by the attacker are not random; they are targeted at system properties and vulnerabilities. Cryptographic integrity mechanisms are effective in detecting random noise events, but they also detect the more systematic deliberate attacks. Cryptographic hash algorithms, such as SHA-1, are designed to make every bit of the hash value a complex, nonlinear function of every bit of the message text, and to make it impractical to find two messages that hash to the same value. On average, it is

[18] Checksum: an algorithm that uses the bits in the transmission to create a checksum value. The checksum value is normally sent in the transmission. The receiver recomputes the checksum value using the bits in the received transmission, and compares the received checksum value with the computed value to determine whether or not the transmission was correctly received. A non-cryptographic checksum algorithm uses a well-known algorithm without secret information (i.e., a cryptographic key).

necessary to perform 2^{80} SHA-1 hash operations to find two messages that hash to the same value, and it is much harder than that to find another message whose SHA-1 hash is the same value as the hash of any given message. Cryptographic message authentication algorithms (MAC) employ hashes or symmetric encryption algorithms and a key to authenticate the source of a message, as well as protect its integrity (i.e., detect errors). Digital signatures use public key algorithms and hash algorithms to provide both authentication and integrity services. Compared to non-cryptographic integrity or authentication mechanisms, these cryptographic services are usually computationally quite expensive; this seems to be unavoidable, since cryptographic protections must resist not just random errors, but deliberate attacks by knowledgeable adversaries with substantial resources.

Cryptographic and non-cryptographic integrity mechanisms may be used together. For example, Consider the TLS protocol (see Part 3). In TLS, a client and a server can authenticate each other, establish a shared "master key" and transfer encrypted payload data. Every step in the entire TLS protocol run is protected by strong cryptographic integrity and authentication mechanisms, and the payload is usually encrypted. Like most cryptographic protocols, TLS will detect any attack or noise event that alters any part of the protocol run. However, TLS has no error recovery protocol. If an error is detected, the protocol run is simply terminated. Starting a new TLS protocol run is quite expensive. Therefore, TLS requires a "reliable" transport service, typically the Internet Transport Control Protocol (TCP), to handle and recover from ordinary network transmission errors. TLS will detect errors caused by an attack or noise event, but has no mechanism to recover from them. TCP will generally detect such errors on a packet-by-packet basis and recover from them by retransmission of individual packets, before delivering the data to TLS. Both TLS and TCP have integrity mechanisms, but a sophisticated attacker could easily fool the weaker non-cryptographic checksums of TCP. However, because of the cryptographic integrity mechanism provided in TLS, the attack is thwarted.

There are some interactions between cryptography and non-cryptographic integrity or error-correction mechanisms that users and protocol designers must take into account. For example, many encryption modes expand ciphertext errors: a single bit error in the ciphertext can change an entire block or more of the resulting plaintext. If forward error correction is applied before encryption, and errors are inserted in the ciphertext during transmission, the error expansion during the decryption might "overwhelm" the error correction mechanism, making the errors uncorrectable. Therefore, it is preferable to apply the forward error correction mechanism after the encryption process. This will allow the correction of errors by the receiving entity's system before the ciphertext is decrypted, resulting in "correct" plaintext.

Interactions between cryptographic and non-cryptographic mechanisms can also result in security vulnerabilities. One classic way this occurs is with protocols that use stream ciphers[19] with non-cryptographic checksums (e.g. CRC-32) and that acknowledge good packets. An attacker can copy the encrypted packet, selectively modify individual ciphertext bits, selectively change bits in the CRC, and then send the packet. Using the protocol's acknowledgement mechanism, the attacker can determine when the CRC is correct, and therefore, determine when

[19] An algorithm that encrypts and decrypts one element (e.g., bit or byte) at a time. There are no FIPS algorithms specifically designated as stream ciphers. However, some of the cryptographic modes defined in [SP 800-38] can be used with a symmetric block cipher algorithm, such as AES, to perform the function of a stream cipher.

the modified packet is acceptable to the protocol. At least one widely used wireless encryption protocol has been broken with such an attack.

APPENDIX B: Key Recovery

Federal agencies have a responsibility to protect the information contained in, processed by and transmitted between their Information Technology systems. Cryptographic techniques are often used as part of this process. These techniques are used to provide confidentiality, assurance of integrity, non-repudiation or access control. Policies **should** be established to address the protection and continued accessibility of cryptographically protected information, and procedures **should** be in place to ensure that the information remains viable during its lifetime. Since cryptographic keying material was used to protect the information, this same keying material may need to be available to remove (e.g., decrypt) or verify (e.g., verify the signature) those protections.

In many cases, the keying material used for cryptographic processes might not be readily available. This might be the case for a number of reasons, including:

1. the cryptoperiod of the key has expired, and the keying material is no longer in operational storage,
2. the keying material has been corrupted (e.g., the system has crashed or a virus has modified the saved keying material in operational storage), or
3. the owner of the keying material is not available, and the owner's organization needs to obtain the protected information.

In order to have this keying material available when required, the keying material needs to be saved somewhere or to be reconstructable (e.g., re-derivable) from other available keying material. The process of re-acquiring the keying material is called key recovery. Key recovery is often used as one method of information recovery when the plaintext information needs to be recovered from encrypted information. However, keying material or other related information may need to be recovered for other reasons, such as the corruption of keying material in normal operational storage (see Section 7.2.1), for the verification of digital signatures for archived documents, or checking the integrity of archived information. Key recovery may also be appropriate for situations in which it is easier or faster to recover the keying material than it is to generate and distribute new keying material. Key recovery is motivated by a need to recover or ascertain the validity of cryptographically protected information (e.g., the information that has been encrypted or authenticated) on behalf of an organization or individual.

However, there are applications that may not need to save the keying material for an extended time because of other procedures to recover an operational capability when the keying material or the information protected by the keying material becomes inaccessible. Applications of this type could include telecommunications where the transmitted information could be resent, or applications that could quickly derive, or acquire and distribute new keying material.

It is the responsibility of an organization to determine whether or not the recovery of keying material is required for their application. The decision as to whether key recovery is required **should** be made on a case by case basis, and this decision **should** be reflected in the Key Management Policy and the Key Management Practices Statement (see Part 2). If the decision is made to provide key recovery, the appropriate method of key recovery **should** be selected, based on the type of keying material to be recovered and the capabilities of the organization, and a suitable key recovery methodology **should** be designed and implemented.

If the decision is made to provide key recovery for a key, all information associated with that key must also be recoverable (see Table 1 in Section 5).

B.1 Recovery from Stored Keying Material

The primary purpose of backing up or archiving keying material is to be able to recover that material when it is not otherwise available in normal operational storage to decrypt or check the information protected by the keying material. For example, encrypted information cannot be transformed into plaintext information if the decryption key is lost or modified; the integrity of data cannot be determined if the key used to verify the integrity of that data is not available. The key recovery process acquires the keying material from backup or archive storage, and places it in normal operational storage, either in the device or module, or in immediately accessible storage (see Section 7.2.1).

B.2 Recovery by Reconstruction of Keying Material

Some keying material may be recovered by reconstructing or re-deriving the keying material from other available keying material, the "base" keying material (e.g., a master key for a key derivation method). The base keying material **shall** be available in either normal operational storage (see Section 7.2.1), backup storage (see Section 7.2.2.1) or archive storage (see Section 7.3.1).

B.3 Conditions Under Which Keying Material Needs to be Recoverable

The decision as to whether to backup or archive keying material for possible key recovery **should** be made on a case by case basis. The decision **should** be based on:

1. the type of key (e.g., signing private key, long-term data encryption key),
2. the application in which the key will be used (e.g., interactive communications, file storage),
3. whether the key is "owned" by the local entity (e.g., a private key) or by another entity (e.g., the other entity's public key) or is shared (e.g., a symmetric data encyption key shared by two entities),
4. the role of the entity in a communication (e.g., sender of receiver),
5. the algorithm or computation in which the key will be used (e.g., does the entity have the necessary information to perform a given computation if the key were to be recovered)[20], and
6. the value of the information protected by the keying material, and the consequences of the loss of the keying material.

The factors involved in a decision for or against key recovery **should** be carefully assessed. The trade-offs are concerned with continuity of operations versus the risk of possibly exposing the keying material and the information it protects if control of the keying material is lost. If it is

[20] This could be the case when performing a key establishment process for some key establishment schemes (see SP 800-56).

determined that a key needs to be recoverable, and the key is lost or modified, but still active (i.e., the cryptoperiod of the key has not expired), then the key **should** be replaced as soon as possible in order to limit its exposure (see Section 7.2.3).

The following subsections provide discussions to assist an organization in determining whether or not key recovery is needed. Although the following discussions address only the recoverability of keys, any related information **should** also be recoverable.

B.3.1 Signature Key Pairs

The private key of a signature key pair (the private signature key) is used by the owner of the key pair to apply digital signatures to information. The associated public key (the public signature verification key) is used by relying entities to verify the digital signature.

B.3.1.1 Public Signature Verification Keys

It is appropriate to backup or archive a public signature verification key for as long as required in order to verify the information signed by the associated private signature key. In the case of a public key that has been certified (e.g., by a Certificate Authority), saving the public key certificate would be an appropriate form of storing the public key; backup or archive storage may be provided by the infrastructure (e.g., by a certificate repository). The public key **should** be stored in backup storage until the end of the public key's cryptoperiod, and **should** be stored in archive storage as long as required for the verification of signed data..

B.3.1.2 Private Signature Keys

Key backup is not usually desirable for the private key of a signing key pair, since the non-repudiability of the signature comes into question. However, exceptions may exist. For example, replacing the private signature key and having its associated public signature verification key distributed (in accordance with Section 7.1.5.1) in a timely manner may not be possible under some circumstances. This may be the case, for example, for the private signature key of a CA. If a private signature key is backed up, the private signature key **shall** be recovered using a highly secure method. Depending on circumstances, the key **should** be recovered for immediate use only, and then **shall** be replaced as soon after the recovery process as possible. Instead of backing up the private signature key, a second private signature key and associated public key could be generated, and the public key distributed in accordance with Section 7.1.5.1 for use if the primary private signature key becomes unavailable. If backup is considered for the private signature key, an assessment **should** be made as to its importance and the time needed to recover the key, as opposed to the time needed to generate a key pair, and certify and distribute a new public signature verification key.

As indicated in Table 4 in Section 7.2.2.1, a private signature key may be reside in backup storage during the key's cryptoperiod; however, a private signature key **shall not** be archived.

B.3.2 Symmetric Authentication Keys

A symmetric authentication key is used to provide assurance of the integrity and source of information. A symmetric authentication key can be used:

(1) by an originator to create a message authentication code (MAC) that can be verified at a later time to determine the authenticity or integrity of the authenticated information; the authenticated information and its MAC could then be stored for later retrieval or transmitted to another entity,

(2) by an entity that retrieves the authenticated information and the MAC from storage to determine the integrity of the stored information (Note: This is not a communication application),

(3) immediately upon receipt by a receiving entity to determine the integrity of transmitted information and the source of that information (the received MAC and the associated authenticated information may or may not be subsequently stored), or

(4) by a receiving and retrieving entity to determine the integrity and source of information that has been received and subsequently stored using the same MAC (and the same authentication key); checking the MAC is not performed prior to storage.

In case 1, the symmetric authentication key need not be backed up if the originator can establish a new authentication key prior to computing the MAC, making the key available to any entity that would need to subsequently verify the information that is authenticated using this new key. If a new authentication key cannot be obtained in a timely manner, then the authentication key **should** be backed up or archived.

In case 2, the symmetric authentication key **should** be backed up or archived for as long as the integrity of the information needs to be determined.

In case 3, the symmetric authentication key need not be backed up if the authentication key can be resent to the recipient. In this case, establishing and distributing a new symmetric authentication key rather than reusing the "lost" key is also acceptable; a new MAC would need to be computed on the information using the new authentication key. Otherwise, the symmetric authentication key **should** be backed up. Archiving the authentication key is not appropriate if the MAC and the authenticated information are not subsequently stored, since the use of the key for both applying and checking the MAC would be discontinued at the end of the key's cryptoperiod. If the MAC and the authenticated information are subsequently stored, then the symmetric authentication key **should** be backed up or archived for as long as the integrity and source of the information needs to be determined.

In case 4, the symmetric authentication key **should** be backed up or archived for as long as the integrity and source of the information needs to be determined.

The symmetric authentication key may be stored in backup storage for the cryptoperiod of the key, and in archive storage until no longer required. If the authentication key is recovered by reconstruction, the "base" key (e.g., the master key for a key derivation method) may be stored in normal operational storage or backup storage for the cryptoperiod of the key, and in archive storage until no longer required.

B.3.3 Authentication Key Pairs

A public authentication key is used by a receiving entity to obtain assurance of the identity of the entity that originated information and the integrity of the information. The associated private authentication key is used by the originating entity to provide this assurance to a receiving entity by computing a digital signature on the information. This key pair may not provide non-repudiation.

B.3.3.1 Public Authentication Keys

It is appropriate to store a public authentication key in either backup or archive storage for as long as required to verify the authenticity of the data that was authenticated by the associated private authentication key. In the case of a public key that has been certified (e.g., by a Certificate Authority), saving the public key certificate would be an appropriate form of storing the public key; backup or archive storage may be provided by the infrastructure (e.g., by a certificate repository). The public key may be stored in backup storage until the end of the private key's cryptoperiod, and may be stored in archive storage as long as required.

B.3.3.2 Private Authentication Keys

When the private key is used only for the authentication of transmitted data, whether or not the authenticated data is subsequently stored, the private authentication key need not be backed up if a new key pair can be generated and the distributed in accordance with Section 7.1.5.1 in a timely manner. However, if a new key pair cannot be generated quickly, the private key **should** be stored in backup storage during the cryptoperiod of the private key. The private key **shall not** be stored in archive storage.

When the private authentication key is used to protect stored information only, the authentication private key **should not** be backed up if a new key pair can be generated. However, if a new key pair cannot be generated, the private key **should** be stored in backup storage during the cryptoperiod of the private key. The private key **shall not** be stored in archive storage.

B.3.4 Symmetric Data Encryption Keys

A symmetric data encryption key is used to protect the confidentiality of stored or transmitted information or both.

Information that is stored needs to be readily available as long as the information is encrypted (i.e., for as long as the information may need to be recovered); this includes information that is both transmitted and stored using the same key. The decryption key (which was also used to encrypt the information) needs to be available during the period of time that the information may need to be recovered. Therefore, the key **should** be backed up or archived during this period.

In order to allow key recovery, the symmetric data encryption key **should** be stored in backup storage during the cryptoperiod of the key, and stored in archive storage after the end of the key's cryptoperiod, if required. In many cases, the key is stored with the encrypted data; the key is wrapped by an archive encryption key or by a symmetric key wrapping key that is wrapped by an archive encryption key.

A symmetric data encryption key that is used for transmission only is used by an originating entity to encrypt information, and by the receiving entity to decrypt the information immediately upon receipt. If the data encryption key is lost or corrupted, and a new data encryption key can be easily obtained by the originating and receiving entities, then the key need not be backed up. However, if the key cannot be easily replaced by a new key, then the key **should** be backed up if the information to be exchanged is of sufficient importance. The data encryption key **should not** be archived when used for transmission only.

B.3.5 Symmetric Key Wrapping Keys

A symmetric key wrapping key is used to wrap (i.e., encrypt) keying material that is to be protected and may be used to protect multiple sets of keying material. The protected keying material is then trans mitted or stored or both.

If a symmetric key wrapping key is used only to transmit keying material, and the key wrapping key becomes unavailable (e.g., is lost or corrupted), it may be possible to either resend the key wrapping key, or to establish a new key wrapping key and use it to resend the keying material. If this is possible within a reasonable timeframe, backup of the key wrapping key is not necessary. If the key wrapping key cannot be resent or a new key wrapping key cannot be readily obtained, backup of the key wrapping key **should** be considered. The archive of a key wrapping key that is only used to transmit keying material is not necessary.

If a symmetric key wrapping key is only used to protect keying material in storage, then the key wrapping key **should** be backed up or archived for as long as the keying material may need to be accessed.

If a symmetric key wrapping key is used for the protection of keying material during both transmission and subsequent storage, then the key wrapping key **should** be backed up or archived for as long as the keying material may need to be accessed.

B.3.6 Random Number Generation Keys

A key used for random (i.e., pseudorandom) number generation **should not** be backed up or archived. If this key is lost or modified, it **shall** be replaced with a new key.

B.3.7 Symmetric Master Keys

A symmetric master key is normally used to derive one or more other keys. It **shall not** be used for any other purpose.

The determination as to whether or not a symmetric master key needs to be backed up or archived depends on a number of factors:

1. How easy is it to establish a new symmetric master key? If the master key is distributed manually (e.g., in smart cards or in hard copy by receipted mail), the master key **should** be backed up or archived. If a new master key can be easily and quickly established using electronic key establishment protocols, then the backup or archiving of the master key may not be desirable, depending on the application.
2. Are the derived keys recoverable without the use of the symmetric master key? If the derived keys do not need to be backed up or archived (e.g., because of their use) or recovery of the derived keys does not depend on reconstruction from the master key (e.g., the derived keys are stored in an encrypted form), then the backup or archiving of the master key may not be desirable. If the derived keys need to be backed up or archived, and the method of key recovery requires reconstruction of the derived key from the master key, then the master key **should** be backed up or archived.

B.3.8 Key Transport Key Pairs

A key transport key pair may be used to transport keying material from an originating entity to a receiving entity during communications, or to protect keying material while in storage. The originating entity in a communication (or the entity initiating the storage of the keying material) uses the public key to encrypt the keying material; the receiving entity (or the entity retrieving the stored keying material) uses the private key to decrypt the encrypted keying material.

B.3.8.1 Private Key Transport Keys

If a key transport key pair is only used during communications, then the private key transport key does not need to be backed up if a replacement key pair can be generated and distributed in a timely fashion. Otherwise, the private key **should** be backed up. Alternatively, one or more additional key pairs could be made available (i.e., already generated and distributed). Archiving the private key is inappropriate.

If the transport key pair is used only during storage, then the private key transport key **should** be backed up or archived for as long as the protected keying material may need to be accessed.

If the transport key pair is used during both communications and storage of keying material, then the private key transport key **should** be backed up or archived for as long as the protected keying material may need to be accessed.

B.3.8.2 Public Key Transport Keys

If the sending entity (the originating entity in a communications) loses the public key transport key or determines that the key has been corrupted, the key can be reacquired from the key pair owner or by obtaining the public key certificate containing the public key (if the public key was certified).

If the entity that applies the cryptographic protection to keying material that is to be stored determines that the public key transport key has been lost or corrupted, the entity may recover in one of the following ways:

1. If the public key has been certified and is stored elsewhere within the infrastructure, then the certificate can be requested.
2. If some other entity knows the public key (e.g., the other entity is the actual owner of the key pair), the key can be requested from this other entity.
3. If the private key is known, then the public key can be recomputed.
4. A new key pair can be generated.

B.3.9 Symmetric Key Agreement Keys

Symmetric key agreement keys are used to establish keying material (e.g., symmetric key wrapping keys, symmetric data encryption keys, or symmetric authentication keys). Each key agreement key is shared between two or more entities. If these keys are distributed manually (e.g., in a key loading device or by receipted mail), then the symmetric key agreement key **should** be backed up. If an electronic means is available for quickly establishing new keys (e.g., a key transport mechanism can be used to establish a new symmetric key agreement key), then a symmetric key agreement key need not be backed up. Symmetric key agreement keys **shall not** be archived.

B.3.10 Static Key Agreement Key Pairs

Static key agreement key pairs are used to establish shared secrets between entities, often in conjunction with ephemeral key pairs (see SP 800-56). Each entity uses their private key agreement key(s), the other entity's public key agreement key(s) and possibly their own public key agreement key(s) to determine the shared secret. The shared secret is subsequently used to derive shared keying material. Note that in some key agreement schemes, one or more of the entities may not have a static key agreement pair (see SP 800-56).

B.3.10.1 Private Static Key Agreement Keys

If the private static key agreement key cannot be replaced in a timely manner, then the private key **should** be backed up in order to continue operations. The private key **should not** be archived unless needed for reconstruction of the keying material.

B.3.10.2 Public Static Key Agreement Keys

If an entity determines that the public static key agreement key is lost or corrupted, the entity may recover in one of the following ways:

1. If the public key has been certified and is stored elsewhere within the infrastructure, then the certificate can be requested.
2. If some other entity knows the public key (e.g., the other entity is the actual owner of the key pair), the key can be requested from this other entity.
3. If the private key is known, then the public key can be recomputed.
4. If the entity is the owner of the key pair, a new key pair can be generated and distributed.

If none of these alternatives are possible, then the public static key agreement key **should** be backed up. The public key **should not** be archived unless needed for reconstruction of the keying material.

B.3.11 Ephemeral Key Pairs

Ephemeral key agreement keys are generated and distributed during a single key agreement process (e.g., at the beginning of a communication session) and are not reused. These key pairs are used to establish a shared secret (often in combination with static key pairs); the shared secret is subsequently used to derive shared keying material. Not all key agreement schemes use ephemeral key pairs, and when used, not all entities have an ephemeral key pair (see SP 800-56).

B.3.11.1 Private Ephemeral Keys

Private ephemeral keys **shall not**[21] be backed up or archived. If the private ephemeral key is lost or corrupted, a new key pair **shall** be generated, and the new public ephemeral key **shall** be provided to the other participating entity in the key agreement process.

B.3.11.2 Public Ephemeral Keys

Public ephemeral keys may be backed up or archived if they are required for reconstruction of the established keying material, and the private ephemeral keys are not required in the key agreement computation.

B.3.12 Symmetric Authorization Keys

Symmetric authorization keys are used to provide privileges to an entity (e.g., access to certain information or authorization to perform certain functions). Loss of these keys will deny the privileges (e.g., prohibit access and disallow performance of these functions). If the authorization key is lost or corrupted and can be replaced in a timely fashion, then the authorization key need not be backed up. A secret authorization key **shall not** be archived.

[21] SP 800-56 states that the ephemeral keys shall be destroyed immediately after use. This implies that the keys **shall not** be backed up or archived.

B.3.13 Authorization Key Pairs

Authorization key pairs are used to provide privileges to an entity. The private key is used to establish the "right" to the privilege; the public key is used to determine that the entity actually has the right to the privilege.

B.3.13.1 Private Authorization Keys

Loss of the private authorization key will deny the privileges (e.g., prohibit access and disallow performance of these functions). If the private key is lost or corrupted and can be replaced in a timely fashion, then the private key need not be backed up. Otherwise, the private key **should** be backed up. The private key **shall not** be archived, since the privilege will not be granted after the end of the cryptoperiod unless a new authorization key pair is provided.

B.3.13.2 Public Authorization Keys

If the authorization key pair can be replaced in a timely fashion (i.e., regeneration of the key pair and secure distribution of the private key to the entity seeking authorization), then the public authorization key need not be backed up. Otherwise, the public key **should** be backed up. The archive of the public key is not appropriate.

B.3.14 Other Cryptographically Related Material

Like keys, other cryptographically related material may need to be backed up or archived, depending on use.

B.3.14.1 Domain Parameters

Domain parameters are used in conjunction with some public key algorithms to generate key pairs (for digital signature generation or verification or for key establishment) or to create and verify digital signatures using those key pairs. The same set of domain parameters is often, but not always) used by a large number of entities.

When an entity (entity A) generates new domain parameters whenever key pairs are generated, these domain parameters are used in subsequent digital signature generation or key establishment processes. The domain parameters need to be provided to other entities that need to verify the digital signatures or with whom keys will be established. If the entity (entity A) determines that its copies of the domain parameters have been lost or corrupted, and if the new domain parameters cannot be securely distributed in a timely fashion, then the domain parameters **should** be backed up or archived. Another entity (entity B) **should** backup or archive entity A's domain parameters until no longer required unless the domain parameters can be otherwise obtained (e.g., from entity A).

When the same set of domain parameters are used by multiple entities, the domain parameters **should** be backed up or archived until no longer required unless the domain parameters can be otherwise obtained (e.g., from a trusted source).

B.3.14.2 Initialization Vectors (IVs)

IVs are used by several modes of operation during the encryption or authentication of information using block cipher algorithms. These IVs **should** be backed up or archived as long as the information protected using those IVs needs to be processed (e.g., decrypted or authenticated).

B.3.14.3 Shared Secrets

Shared secrets are generated by each entity participating in a key agreement process. The shared secret is then used to derive the shared keying material to be used in subsequent cryptographic operations. Shared secrets may be generated during interactive communications (e.g., where both entities are online) or during non-interactive communications (e.g., in store and forward applications).

A shared secret **shall not** be backed up or archived.

B.3.14.4 Secret Seeds

Secret seeds are used in the generation of pseudorandom random numbers that need to remain secret. These seeds are not shared with other entities and **shall not** be reused. Therefore, secret seeds **shall not** be backed up or archived.

B.3.14.5 Public Seeds

Public seeds are used in the generation of pseudorandom numbers that may be validated (e.g., in order to validate domain parameters). The public seed **should** be backed up or archived as long as these numbers need to be validated.

B.3.14.6 Other Public Information

Public information (e.g., a nonce) is often used during key establishment. The public information may need to be available to process the cryptographically protected information (e.g., to decrypt or authenticate); therefore, the public information **should** be backed up or archived until no longer needed to process the protected information.

B.3.14.7 Intermediate Results

The intermediate results of a cryptographic operation **shall not** be made backed up or archived. The cryptographic operations **should** be re-initialized or restarted.

B.3.14.8 Key Control Information

Key control information is used, for example, to determine the keys and other information to be used to process cryptographically protected information (e.g., decrypt or authenticate), to identify the purpose of a key, or the entities that share the key (see Section 5.2.3).

Key control information **should** be backed up or archived for as long as the associated key needs to be available.

B.3.14.9 Random Numbers

Random numbers are generated by random number generators. The backup or archiving of a random number depends on how it is used.

B.3.14.10 Passwords

A password is used to acquire access to privileges by an entity. The loss of a password will deny the privileges. If the password is lost or corrupted and can be replaced in a timely fashion, then the password need not be backed up. A password **shall not** be archived.

B.3.14.11 Audit Information

Audit information containing key management events **shall** be backed up and archived.

B.4 Key Recovery Systems

Key recovery is a broad term that may be applied to several different key recovery techniques. Each technique will result in the recovery of a cryptographic key and other information associated with that key (i.e., the keying material). The information required to recover that key may be different for each application or each key recovery technique. The term “Key Recovery Information” (KRI) is used to refer to the aggregate of information that is needed to recover or verify cryptographically protected information. Information that may be considered as KRI includes the keying material to be recovered or sufficient information to reconstruct the keying material, other associated cryptographic information, the time when the key was created, the identity of the owner of the key (the individual, application or organization who created the key or who own the data protected by that key) and any conditions that must be met by a requestor to be able to recover the keying material.

When an organization determines that key recovery is required for all or part of its keying material, a secure Key Recovery System (KRS) needs to be established in accordance with a well defined Key Recovery Policy (see Appendix B.5). The KRS **should** support the Key Recovery Policy and consists of the techniques and facilities for saving and recovering the keying material, the procedures for administering the system, and the personnel associated with the system.

When key recovery is determined to be necessary, the KRI may be stored either within an organization (in backup or archive storage) or may be stored at a remote site by a trusted entity. There are many acceptable methods for enabling key recovery. A KRS could be established using a safe for keying material storage; a KRS might use a single computer that provides the initial protection of the plaintext information, storage of the associated keying material and recovery of that keying material; a KRS may include a network of computers with a central Key Recovery Center; or a KRS could be designed using other configurations. Since a KRS provides an alternative means for recovering cryptographic keys, a risk assessment **should** be performed to ensure that the KRS adequately protects the organization’s information and reliably provides the KRI when required. It is the responsibility of the organization that needs to provide key recovery to ensure that the Key Recovery Policy, the key recovery methodology, and the Key Recovery System adequately protect the KRI.

A KRS **should**:

1. Generate or provide sufficient KRI to allow recovery or verification of protected information.
2. Ensure the validity of the saved key and the other KRI.
3. Ensure that the KRI is stored with persistence and availability that is commensurate with that of the corresponding cryptographically protected data.
4. Any cryptographic modules used by the KRS **shall** be compliant with FIPS 140-2.
5. The KRS **shall** use FIPS-approved or NIST recommended algorithms, when cryptography is used.
6. The strength of any algorithms used to protect KRI **shall** be commensurate with the sensitivity of the information associated with the KRI.
7. The KRS **shall** be designed to enforce the Key Recovery Policy (see Section B.5).

8. The KRS **shall** protect KRI against unauthorized disclosure or destruction. The KRS **shall** verify the source of requests and ensure that only requested and authorized information is provided to the requestor.
9. The KRS **shall** protect the KRI from modification.
10. The KRS **shall** have the capability of providing an audit trail. The audit trail **shall not** contain the keys that are recovered or any passwords that may be used by the system. The audit trail might include the identification of the event being audited, the time of the event, the identity of the user causing the event, and the success or failure of the event.
11. The KRS **shall** limit access to the KRI, the audit trail and authentication data to authorized individuals.
12. It **should not** be possible to modify the audit trail.

B.5 Key Recovery Policy

An organization that determines that key recovery is required for some or all of their keying material should develop a Key Recovery Policy that addresses the protection and continued accessibility of that information. For each system, application and cryptographic technique used, consideration must be given as to whether or not the keying material may need to be saved for later recovery of the keying material to allow subsequent decryption or checking of the information protected by the keying material.

The policy **should** address (at a minimum):

1. The keying material that needs to be saved for a given application. For example, keys and IVs used for the decryption of stored information protected by the keys and IVs may need to be saved. Keys for the authentication of stored or transmitted information may also need to be saved.
2. How and where the keying material would be saved? For example, the keying material could be stored in a safe by the individual who initiates the protection of the information (e.g., the encrypted information), or the keying material could be saved automatically when the protected information is transmitted, received or stored. The keying material could be saved locally or at some remote site.
3. Who will be responsible for protecting the KRI? Each individual, organization or sub-organization could be responsible for their own keying material, or an external organization could perform this function.
4. Who can request key recovery and under what conditions? For example, the individual who protected the information (i.e., used and stored the KRI) or the organization to which the individual is assigned could recover the keying material. Legal requirements may need to be considered. An organization could request the information when the individual who stored the KRI is not available.
5. Under what conditions could the policy be modified and by whom?
6. What audit capabilities and procedures would be included in the KRS? The policy **should** identify the events to be audited. Auditable events might include KRI requests and their associated responses; who made a request and when; the startup and shutdown of audit

functions; the operations performed to read, modify or destroy the audit data; requests to access user authentication data; and the uses of authentication mechanisms.

7. How the KRS would deal with aged keying material or the destruction of the keying material.

8. Who would be notified when keying material is recovered and under what conditions? For example, the individual who encrypted data and stored the KRI could be notified when the organization recovers the decryption key because the person is absent, but the individual might not be notified when the organization is monitoring the activities of that individual.

9. The procedures that need to be followed when the KRS or some portion of the data within the KRS is compromised.

APPENDIX C: Keys Generated from Passwords

[Note: The following discussion has been provided to invite comment on the use of passwords for the generation of cryptographic keys. Some or all of this material may be provided in another document at a later time. However, at the present time, this material should not be considered as the NIST position on the subject.]

A password is a secret character string that a user is expected to memorize. Passwords are used in the implementation of many security services. Typically, passwords are an authentication and access control mechanism; a party who knows an authorized name and password pair is granted access to a system, an account, a directory or a file. But the mechanisms for accomplishing the authentication from the password may be cryptographic, and in many cases, the password is effectively serving as a cryptographic key, even a key wrapping key.

A key is said to be generated from a password when the password is the only source of entropy, the only unpredictable data, from which the key is generated. Thus, only the password and some general key generation method to generate the key needs to be known. This differs from "non-password" based deterministic key generation processes in which the random number generation process is typically "seeded" with some large value that is chosen to be highly unpredictable.

Some two party protocols (see Appendix C.7) use a shared password as a part of a cryptographic process that generates a high quality shared key, but the key generation process depends on conventional random numbers, not the password, and an eavesdropper does not learn anything about the password or the key. In this case, the shared key is not generated from the password, since it is not the primary source of entropy in the key, however possession of the generated key can still be used to authenticate the parties, or to protect an encrypted session.

Using a key generated a key from a password can be risky because a password with the strength of a 128-bit or even an 80-bit cryptographic key is very difficult, if not impossible, to memorize. Appendix C.1 discusses the required length for randomly chosen passwords that would match the strength of randomly generated cryptographic keys. However, random strings are hard to memorize, and if the passwords are chosen to be easily memorized, then they must be even longer to have an equivalent strength.

Therefore, keys generated directly from passwords are likely to be a weak point in any cryptographic application. Rules intended to enforce the use of "strong" passwords (long random strings) may cause users to write down their passwords, rather than memorize them, and keep the written copy in convenient but insecure locations, which makes any cryptographic protection vulnerable, however strong it may otherwise be. Nevertheless, it is very common to generate keys directly from passwords in many applications, and useful protection can sometimes be obtained if passwords are well chosen, systems and applications are properly designed, and, most importantly, users are disciplined in how they, slelect, keep and use their passwords.

C.1 Passwords Equivalent to Cryptographic Keys

Table 10 shows how long a completely random password needs to be to have approximately the same strength as a cryptographic key, depending on the size of the alphabet used to generate the password.

Table 10: Key Size/Password Size Equivalence

Key bits	10 decimal digits	26 letters	36 letters & numbers	94 keyboard characters	1000 word dictionary
56	17	12	11	9	6
64	20	13	13	10	7
80	25	17	16	12	8
112	34	24	22	17	11
128	39	27	25	20	13

Column 1 shows symmetric cryptographic keys of commonly used sizes. Column 2 shows the number of decimal digits required to have approximately the same number of possible values as the first column. Column 3 shows the corresponding number of characters required if the password uses the 26 roman letters, column 4 uses the 26 letters plus the ten digits, and column 5 assumes that all 94 printable characters of a computer keyboard are used. The rightmost column assumes that words are chosen at random from a 1000 word dictionary of ordinary words.

All these equivalences assume that the passwords are chosen entirely at random, and was generated by finding the number of characters required to have about the same number of possible values as the corresponding cryptographic key. In all the columns but the rightmost column, the length that produced the nearest equivalence was selected; sometimes the key is a little stronger and sometimes the password is a little stronger. In the right most column, the alphabet is so large that the difference in strength to the nearest equivalent could be a factor of 100 or more, and therefore, the password length was always chosen to exceed the key strength. Obviously, a password of 39 randomly chosen decimal digits, 20 randomly characters from a 94 character alphabet, or 13 randomly chosen English words is extremely difficult to memorize, yet this is what is needed to match the strength of a good 128-bit AES key. Even a password equivalent to a 56-bit DES key is challenging to remember.

Almost anything that is done make the password easier to remember is going to make it weaker, that is, to reduce the amount of computation needed to attack the password by exhaustion of all possible passwords.

C.2 Attacks on Passwords

Passwords are subject to guessing, dictionary and exhaustion attacks. In a guessing attack the adversary makes guesses about the password based on some knowledge of the user, since users frequently base their passwords on things like their phone numbers, middle name, date of birth and the like, as an aid to memory, and the attacker may also know these things. Dictionary attacks rely on the fact that common words and names are often used in passwords. Dictionaries of millions of words and variations on words exist for such attacks, and are ordered so that the most likely values are tried first. Exhaustion attacks simply systematically try the entire possible password space, usually after the dictionary attack has failed. Exhaustion attacks can be ordered, so that shorter or more likely passwords are tried first.

Password attacks can be on-line and off-line attacks. An on-line attack is generally done with a server, a device or an application program, by pretending to be a user and submitting a number of authentication attempts (i.e., submit a number of password guesses). For example, the attacker tries to log-on to a remote computer system across a network by submitting username/password pairs. In an off-line attack, the attacker obtains all the information that is needs to conduct the attack on the attacker's own computer, using the attacker's own software. The attacker intercepts an encrypted message and tries to decrypt it on his own computer; if he can find the right encryption key, he will presumably be able to recognize the decrypted message (for example, because it becomes English words).

Off-line attacks are generally much more powerful than on-line attacks. There are no constraints or limitations on the attacker in an off-line attack, while on-line attacks can often be detected and thwarted (by example by locking an account after a certain number of unsuccessful logon attempts), or the rate at which they can be run can be limited by servers, even if the attacks are automated. But no such limitations are possible for off-line attacks.

C.3 Password Strength

It is useful to class passwords into three general categories, by their strength or resistance to attacks, as defined in Table 11: weak, strong and inexhaustible.

Table 11: Password Strength Classes

Password Type	Characterization	Attack Protection	Example
Weak	Resists a small number of guesses	Protection against limited on-line guessing, when combined with a token such as an ATM card, or a biometric	Four digit PIN
Strong	Resistant to multiple guesses by a human	Protects against on-line exhaustion	Eight or more alphanumeric and special character password
Inexhaustible	Resistant to multiple guesses by a computer	Protection against off-line exhaustion by computers	Fifteen or more alphanumeric and special character password and passphrase

Weak passwords are very common, often in the form of a 4 to 6 digit Personal Identification Number, or PIN. When PINs are used with a bank card, at an automatic teller machine, for example (often in a transaction that is videotaped), sufficient security can be achieved if the teller machine protocol will limit the number of attempts to use the card, and the bank can monitor the pattern of use of the card. A pin or weak password may also, perhaps, provide a useful degree of security when it is used to activate a cryptographic token, if the token is also designed to lock or erase the key after a set number of unsuccessful activation attempts. The intruder must steal the token, then attempt to guess the key.

However, where the weak password is the only authentication factor (e.g., imagine an ATM transaction where no card was needed, just a PIN), where an attacker is free to conduct repeated on-line attacks or attacks against a broad range of targets, and where that attack can be automated in some manner, then weak passwords offer very little protection.

Weak passwords, simply do not make good sources of cryptographic keys; cryptographic mechanisms are intended to provide at least some resistance to offline attacks, where the attacker has all the information needed to conduct an attack on a machine of his own choice. For example, consider an access control protocol where a system sends a challenge to a user and the user cryptographically combines a password and the challenge to generate a response. An eavesdropper who captures the challenge and reply can quickly try all the values of a 4, 5 or 6 digit PIN and determine the PIN.

Inexhaustible passwords, on the other hand, are equivalent to cryptographic keys, and it is possible that keys could be generated from them if done properly. A completely random 12-character string of printable letters, numbers and special characters has roughly as many possible values as an 80-bit cryptographic key. A 12 character, truly random string, however, is extremely difficult to memorize, so two realistic alternatives exist:

1. The random string is written and kept secured until it is needed, that is, locked in a safe or some other secure place. Obviously, this is cumbersome if good security is provided, but may be workable for situations where the password is used only occasionally.
2. The string is chosen not at random, but to facilitate memory. If it is to be as difficult to attack, such a string is necessarily longer than if it were chosen randomly, but may be easier to memorize than a shorter random string (see Appendix C.4).

Creating and memorizing good inexhaustible passwords requires thought and effort, and is more an art than a science. However, when a key is generated from a well-chosen inexhaustible password, an eavesdropping attack on a challenge-response protocol will fail with a very high probability. If a file is encrypted with a strong algorithm under a key generated from an inexhaustible password, then an intruder who steals the ciphertext file will not be able to recover the plaintext with an offline password exhaustion attack.

Weak passwords provide little protection when used alone, and inexhaustible passwords are so hard to generate, remember and use that they must be reserved for the most important cases. However, most of us today need to use many passwords. Strong passwords fall in the middle, between weak and inexhaustible passwords; they can, in principle be attacked by off-line exhaustion, but not without a great deal of work. Strong passwords are strong enough to resist on-line attacks in any well-designed application, and they are not as difficult to remember as inexhaustible passwords. Strong passwords:

a. are at least 8 characters long;
b. contain at least one upper case letter, one lower case letter, one numeric and one special character;
c. are not based on some characteristic or personal information of the user, such as his social security number, his street address or phone number, the name of his wife, etc.
d. avoid the inclusion of common words or names in the password.

It is recommended that weak passwords only be used (to generate keys or in other ways) when they are accompanied by other protection factors, such as tokens or biometrics, to provide adequate security. Weak passwords must not be used in processes, protocols or applications that are susceptible to off-line attacks. Strong or inexhaustible passwords must be used whenever a password is the only protection factor against an attack, and whenever offline attacks are possible. When highly sensitive data are encrypted under a key generated from a password and the ciphertext is exposed or vulnerable to theft, (e.g., the cipertext is transmitted over a network, or is kept in a laptop or PDA that is readily exposed to theft) then an inexhaustible password **shall** be used.

C.4 Creating Strong or Inexhaustible Passwords that Can be Remembered

Creating long strings that appear to be arbitrary and random, but actually have some structure that facilitates memory, depends on the mind of the password user. There are many techniques that are used, and guidance in this area is often more what not to do than what to do (e.g., don't use the names or nicknames of sports teams, don't use your dog's name, don't use your social security number, etc.).

Attempts to create memorable inexhaustible passwords are often based on phrases or sentences. One technique is to simply pick a memorable sentence or phrase and use the first letter of each word. "I am the Captain of the Pinafore, and a right good Captain too." becomes "IatCotPaargCt." This is a 14-character string, but is weaker than a truly random 14-character string. Obviously, the string includes only the 52 alphabetic characters plus 1 special character, lower case letters occur more frequently than upper case letters, and the resulting password reflects the frequency distribution of English letters. Another method simply uses the whole phrase, for example: "IamtheCaptainofthePinafore". We are up to 26 characters, but now the password string consists entirely of English words and, again, reflects the frequency distribution of letters in English. A better passphrase might be: "I^am^the^Capitan^of^the^Pina4". This is up to 29 characters, and includes both special characters and a number. A better (but harder to remember) version still would be: "I+am_the)Captain(of*the&Pina4". But any of these would probably defeat all but the most determined attacks, except, perhaps, an attack inspired by the insight that the password holder is a Gilbert and Sullivan buff.

But whatever technique is used, the technique usually depends on something well known to the password holder - some favorite text, some obscure interest or hobby, etc. - and then a rule can be applied to produce an apparently arbitrary string.

C.5 Passwords as Keys in Challenge-Reply Authentication Protocols

Passwords are often used as secrets or keys in a challenge-reply authentication protocol. In such a protocol, the password and a random challenge are input into a cryptographic function (e.g., the password and challenge might be concatenated and hashed with SHA-1. The output then becomes the reply. There are many variations, such as the APOP protocol [RFC1939], S-KEY [RFC1760] and others.

These challenge-reply techniques are sometimes said to use a "one-time password" because the reply is not used again. However, this is a misnomer; the challenge-reply protocol provides protection against the unsophisticated eavesdropper, but does not provide good protection against the more sophisticated attacker. Challenge-reply methods are more secure than sending a password across a network in an unencrypted message, but are usually vulnerable to eavesdroppers and off-line attacks.

These protocols usually can be attacked offline by an eavesdropper who records the protocol run, even where cryptographic challenge-reply authentication protocols are used. The eavesdropper intercepts the challenge and the reply, then conducts an offline dictionary or exhaustion attack on the pair, trying different passwords with the challenge until the reply is found. Depending on the strength of the password, the attacker is quite likely to succeed, however strong the cryptographic functions employed.

C.6 Passwords Used to Generate Encryption Keys

There are many commercial products that use a password to generate an encryption key (e.g., by hashing the password, and then use that key to encrypt a file or another key). **In general, these products have not been evaluated, and NIST currently does not provide guidance on the generation of keys from passwords.** [PKCS #5] is a de facto industry standard for generating keys from passwords. The user then supplies the same password to decrypt the file or activate the key by decrypting the file. If the ciphertext file is stolen, a "password encrypted" file is inherently subject to off-line password attacks, however strong the encryption algorithm. Alternatively, in some cases a file may be encrypted under a randomly generated key, and the randomly generated encryption key is encrypted under a key generated from a password; the key is, in effect, encrypted by the password. If the password encrypted key is kept separately from the encrypted data and well secured, then this approach may be more secure, because an intruder has to steal both the encrypted key and the encrypted data to conduct an attack.

Keys stored in cryptographic modules are often protected by encrypting them in a key derived from a password. The password is, in effect, a key encrypting key. This can provide fairly good protection, since the module can be designed to frustrate a guessing or dictionary attack. When keys derived from passwords are used to encrypt other keys, it may be preferable, in some cases, to not include a checksum with a key encrypted under a key derived from a password, because such a checksum might facilitate an offline attack.

Keys encrypted under a key that is generated from a password and all the files that those encrypted keys protect may be vulnerable to theft, followed by off-line attacks. It is vital, then, that adequate physical security be provided, and the password used to protect these keys **should** be as strong as possible.

C.7 Strong Keys from Passwords and Servers

Cryptographic techniques are available for communications sessions that allow a server and client to create a symmetric key from a shared password (e.g., [SPEKE], [EKE] and [P1363.2]). In several of these techniques, the password is used as a component of a Diffie-Hellman key agreement operation, along with other random factors used in the key agreement. The entropy of the final resulting key depends not on the password, but on the random factors. However, if the client and server use the same password, they will generate the same strong symmetric key at both the client and server, otherwise they will not. This symmetric key can then be used for either encryption, or authentication, or both. This key is not really "generated" from a password, because the password is not the primary source of entropy in the key. However, to the user, the key appears to have been generated from the password. Note that, while the key may be strong in terms of entropy, the authentication of the parties to each other is only as strong as the shared password against an on-line guessing attack. Thus, a weak key **should not** be used in this protocol. **Also note that these techniques have not been evaluated, and NIST currently does not, in general, provide guidance on them.**

An eavesdropper, who intercepts the protocol run, will not be able to conduct an offline password attack. Therefore, these protocols are superior to password-based challenge/reply protocols for authentication or for generating confidentiality session keys, because they may not be susceptible to off-line password attacks.

In an alternative strong password technique, one of the two random factors of an RSA private key may be generated as needed from a password supplied by a user [PWENPKI][2PTYSIG]. The other factor is generated randomly on a server. The RSA public key is generated by the server from both factors. In this scheme, RSA encryption or decryption operations require the cooperation of the server for each operation; the client does "half" the operation using a key generated by a password, and the server does the other half. Neither the server nor the client can perform the operation alone. A relying party uses the RSA public key just as he would use any other RSA key. The key is not truly generated by the password, rather the key is derived from a conventional random number generation processes.

With the methods described above, a secure server participates in all operations. The server contributes a random component, the client may also contribute a large random component, and while the password is used in the key generation process, an eavesdropper who intercepts the messages cannot conduct an offline attack against the password. An eavesdropper has as difficult a problem as an attack on a conventional Diffie-Hellman or RSA operation.

These protocols can be used for client server authentication (e.g., to authenticate an e-mail client to a POP server. The RSA-based technique can be used for any normal RSA operation. At the present time, these techniques are used primarily in PKI applications (e.g., to download a private key from a server to a client); however, these techniques can potentially be used wherever a password is used to authenticate to a server.

A disadvantage of the Diffie-Hellman based strong password techniques is that the server must retain the true passwords themselves, rather than an inversion of the passwords (this is also true of password challenge/reply protocols like APOP). These passwords must be strongly protected. Nevertheless, when passwords are employed in authentication protocols, or for session encryption, these strong password techniques may significantly improve the security of password-based authentication and encryption, even when users choose relatively weak passwords, by eliminating off-line password attacks.

APPENDIX X: References

[AC]	Applied Cryptography, Schneier, John Wiley & Sons, 1996.
[ANSI-X9.31]	Digital Signatures Using reversible Public Key Cryptography for the Financial Services Industry (rDSA), 1998
[ANSI-X9.42]	Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using discrete Logarithm Cryptography, 2001
[ANSI-X9.44]	Public Key Cryptography for the Financial Services Industry: Key Agreement Using Factoring-Based Cryptography, [Insert Date]
[ANSI-X9.52]	Triple Data Encryption Algorithm Modes of Operation, July, 1998
[ANSI-X9.62]	Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1998
[ANSI-X9.63]	Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, [Insert Date]
[FIPS46-3]	Data Encryption Standard (DES), October 25, 1999
[FIPS140-2]	Security Requirements for Cryptographic Modules, May 25, 2001
[FIPS180-2]	Secure Hash Standard (SHS), August 2002.
[FIPS186-2]	Digital Signature Standard (DSS), June 2000.
[FIPS197]	Advanced Encryption Standard (AES), November 2001.
[FIPS198]	Keyed-Hash Message Authentication Code (HMAC), [Insert Date]
[SP 800-56]	[Key Establishment Schemes Standard], [Insert date]
[HAC]	Handbook of Applied Cryptography, Menezes, van Oorschot and Vanstone, CRC Press, 1996.
[IPKI]	Introduction to Public Key Cryptography and the Federal PKI Infrastructure; Kuhn, Hu, Chang and Polk; draft NIST Special Publication, 2001 [available at http://csrc.nist.gov/publications]
[ITL Bulletin]	Techniques for System and Data Recovery, NIST ITL Computer Security Bulletin, April 2002

[SP 800-38]	Special Publication 800-38, Recommendation for Block Cipher Modes of Operation, December 2001.
[PKCS1]	PKCS #1 v2.0: RSA Cryptography Standard, RSA Laboratories, October, 1998.
[SP800-5]	A Guide to the Selection of Anti-Virus Tools and Techniques, December 1992
[OMB11/01]	OMB Guidance to Federal Agencies on Data Availability and Encryption, Office of Management and Budget, 26 November 2001.
[SPEKE]	D. Jablon, "The SPEKE Password-Based Key Agreement Methods", available at http://www.ietf.org/internet-drafts/draft-jablon-speke-01.txt, April 15, 2002.
[EKE]	S. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-based protocols secure against dictionary attacks," in IEEE Symposium on Research in Security and Privacy, pp. 72-84, May 1992.
[P1363.2]	IEEE P1363.2, "Standard Specifications for Public-Key Cryptography: Password-Based Techniques," (work in progress), IEEE, available at http://grouper.ieee.org/groups/1363/.
[PWENPKI]	Ravi Sandhu, Mihir Bellare and Ravi Ganesan, "Password-Enabled PKI: Virtual Smartcards versus Virtual Soft Tokens," available at http://www-singlesignon.net/crypto_science_paper.htm.
[2PTYSIG]	Mihir Bellare and Ravi Sandhu, "The Security of Practical Two-Party RSA Signature Schemes," available at http://www.cse.ucsd.edu/users/mihir/papers/splitkey.html.
[RFC1760]	N. Haller, "The S/KEY One-Time Password System."
[RFC1939]	J. Myers and M. Rose, "Post Office Protocol - Version 3."