Publication Number: **NIST Special Publication (SP) 800-57 Part 1 Revision 4**

Title: **Recommendation for Key Management, Part 1: General**

Publication Date: **1/28/2016**

- Final Publication: http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4 (which links to http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf).
- Related Information on CSRC: http://csrc.nist.gov/publications/PubsSPs.html#SP-800-57-Part%201-Rev.%204 and http://csrc.nist.gov/groups/ST/toolkit/key_management.html
- Information on other NIST cybersecurity publications and programs can be found at: http://csrc.nist.gov/

The following information was posted with the attached DRAFT document:

Sep. 10, 2015

SP 800-57 Part 1-Rev. 4

DRAFT Recommendation for Key Management: Part 1: General (Revision 4)

NIST requests comments on a revision of Special Publication (SP) 800-57, Part 1, *Recommendation for Key Management, Part 1 (Rev. 4)*. This Recommendation provides general guidance and best practices for the management of cryptographic keying material. A list of changes is provided in Appendix D of the document. Please send comments to keymanagement @nist.gov by **October 31, 2015**, with "Comments on SP 800-57, Part 1" in the subject line.

DRAFT NIST Special Publication 800-57, Part 1, Rev. 4

Recommendation for Key Management – Part 1: General (Revision 4)

Elaine Barker

http://dx.doi.org/10.6028/NIST.SP.XXX

C O M P U T E R   S E C U R I T Y

National Institute of Standards and Technology
U.S. Department of Commerce

DRAFT NIST Special Publication 800-57, Part 1, Rev. 4

Recommendation for Key Management – Part 1: General (Revision 4)

Elaine Barker
Computer Security Division
Information Technology Laboratory

http://dx.doi.org/10.6028/NIST.SP.XXX

September 2015

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie E. May, Under Secretary of Commerce for Standards and Technology and Director

Authority

This publication has been developed by NIST to further its statutory responsibilities under the Federal Information Security Management Act (FISMA), Public Law (P.L.) 107-347. NIST is responsible for developing information security standards and guidelines, including minimum requirements for Federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate Federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), *Securing Agency Information Systems*, as analyzed in Circular A-130, Appendix IV: *Analysis of Key Sections*. Supplemental information is provided in Circular A-130, Appendix III, *Security of Federal Automated Information Resources*.

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

National Institute of Standards and Technology Special Publication 800-90A
Natl. Inst. Stand. Technol. Spec. Publ. 800-57, art 1, 168 pages (September 2015)
http://dx.doi.org/10.6028/NIST.SP.XXX
CODEN: NSPUE2

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by Federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, Federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. All NIST Computer Security Division publications, other than the ones noted above, are available at http://csrc.nist.gov/publications.

Public comment period: September 11, 2015 to October 31, 2015

All comments are subject to release under the Freedom of Information Act (FOIA)

National Institute of Standards and Technology

Attn: Computer Security Division, Information Technology Laboratory

100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930

Email: keymanagement@nist.gov

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in Federal information systems. The Special Publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

Abstract

This Recommendation provides cryptographic key management guidance. It consists of three parts. Part 1 provides general 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. Finally, Part 3 provides guidance when using the cryptographic features of current systems.

Keywords

archive; assurances; authentication; authorization; availability; backup; compromise; confidentiality; cryptanalysis; cryptographic key; cryptographic module; digital signature; hash function; key agreement; key management; key management policy; key recovery; key transport; originator-usage period; private key; public key; recipient-usage period; secret key; split knowledge; trust anchor.

Acknowledgements

The National Institute of Standards and Technology (NIST) gratefully acknowledges and appreciates contributions by previous authors of this document on the many security issues associated with this Recommendation: William Barker, William Burr, and Timothy Polk from NIST; Miles Smid from Orion Security; and Lydia Zieglar from the National Security Agency. NIST also thanks the many contributions by the public and private sectors whose thoughtful and constructive comments improved the quality and usefulness of this publication.

Overview

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 a safe combination is 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 the keys, and the protection afforded to the keys. 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, use and destruction of keys.

Users and developers are presented with many 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. This Recommendation (i.e., SP 800-57) provides background information and establishes frameworks to support appropriate decisions when selecting and using cryptographic mechanisms.

This Recommendation does not address the implementation details for cryptographic modules that may be used to achieve the security requirements identified. These details are addressed in Federal Information Processing Standard (FIPS) 140 [FIPS 140], the associated implementation guidance and the derived test requirements (available at http://csrc.nist.gov/groups/STM/cmvp/standards.html).

This Recommendation is written for several different audiences and is divided into three parts.

Part 1, *General*, contains basic key management guidance. It is intended to advise developers and system administrators on the "best practices" associated with key management. Cryptographic module developers may benefit from this general guidance by obtaining a greater understanding of the key management features that are 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 1 of the Recommendation:

1. Defines the security services that may be provided and key types that may be employed in using cryptographic mechanisms.
2. Provides background information regarding the cryptographic algorithms that use cryptographic keying material.
3. 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.
4. Identifies the states in which a cryptographic key may exist during its lifetime.
5. Identifies the multitude of functions involved in key management.

6. Discusses a variety of key management issues related to the keying material. Topics discussed include key usage, cryptoperiod length, domain-parameter validation, public-key validation, accountability, audit, key management system survivability, and guidance for cryptographic algorithm and key size selection.

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 the key management aspects of statutory and policy security planning requirements for Federal government organizations.

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

Table of Contents

Tables

Figures

RECOMMENDATION FOR KEY MANAGEMENT

Part 1: General

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) and NIST Recommendations (which are published as Special Publications) that specify cryptographic techniques for protecting sensitive, unclassified information.

Since NIST published the Data Encryption Standard (DES) in 1977, the suite of **approved** standardized algorithms has been growing. New classes of algorithms have been added, such as secure hash functions 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. This NIST Recommendation applies to U.S. government agencies using cryptography for the protection of their sensitive, unclassified information. This Recommendation 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 is known by 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 the 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 unauthorized substitution and modification. 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.

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. This Recommendation (i.e., SP 800-57) provides background information and establishes frameworks to support appropriate decisions when selecting and using cryptographic mechanisms.

1.2 Audience

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

Part 1 of this Recommendation 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 the key management features that are 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 Recommendation to determine which configuration settings are most appropriate for their information.

Part 2 of this Recommendation [SP800-57, Part 2] 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 this Recommendation addresses the key management issues associated with currently available cryptographic mechanisms and is intended to provide guidance to system installers, system administrators and end users of existing key management infrastructures, protocols, and other applications, as well as the people making purchasing decisions for new systems using currently available technology.

Though some background information and rationale are provided for context and to support the recommendations, this document assumes that the reader has a basic understanding of cryptography. For background material, readers may look to a variety of NIST and commercial publications, including [SP800-32], which provides an introduction to a public-key infrastructure.

1.3 Scope

This Recommendation 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.

This Recommendation 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 Recommendation. Some of the topics noted above are addressed in other NIST standards and guidance. This Recommendation supplements more-focused standards and guidelines.

This Recommendation does not address the implementation details for cryptographic modules that may be used to achieve the security requirements identified. These details are addressed in [FIPS140], the FIPS 140 implementation guidance and the derived test requirements (available at http://csrc.nist.gov/ groups/STM/cmvp/standards.html).

This Recommendation 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 Recommendation 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 an important recommendation. Ignoring the recommendation could result in undesirable results. Note that **should** may be coupled with **not** to become **should not**.

1.4 Purpose of FIPS and NIST Recommendations (NIST Standards)

Federal Information Processing Standards (FIPS) and NIST Recommendations, collectively referred to as "NIST standards," are valuable because:

1. They establish an acceptable minimal level of security for U.S. government systems. Systems that implement these NIST standards offer a consistent level of security **approved** for the protection of sensitive, unclassified government data.

2. They often establish some level of interoperability between different systems that implement the NIST 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. They often provide for scalability, because the U.S. government requires products and techniques that can be effectively applied in large numbers.

4. They are scrutinized by U.S. government experts and the public to ensure that they provide a high level of security. The NIST standards process invites broad public participation, not only through the formal NIST public review process before adoption, but also by interaction with the open cryptographic community through NIST workshops, participation in voluntary standards development organizations, participation in cryptographic research conferences and informal contacts with researchers. NIST encourages study and cryptanalysis of NIST Standards, and inputs on their security are welcome at any point, from initial requirements, during development and after adoption.

5. NIST-**approved** cryptographic techniques are periodically re-assessed for their continued effectiveness. If any technique is found to be inadequate for the continued protection of government information, the NIST standard is revised or discontinued.

6. The algorithms specified in NIST standards (e.g., AES, TDEA, SHA-1, and DSA) and the cryptographic modules in which they reside have required conformance tests. These tests are performed by accredited laboratories on vendor implementations that claim conformance to the standards. Vendors are permitted to modify non-conforming implementations so that they meet all applicable requirements. Users of validated implementations can have a high degree of confidence that validated implementations conform to the standards.

Since 1977, NIST has developed a cryptographic "toolkit" of NIST standards[1] that form a basis for the implementation of **approved** cryptography. This Recommendation references many of those standards, and provides guidance on how they may be properly used to protect sensitive information.

1.5 Content and Organization

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 *Recommendation for Key Management*
2. Section 2, *Glossary of Terms and Acronyms*, provides definitions of terms and acronyms used in this part of the *Recommendation for Key Management*. The reader should be aware that the terms used in this Recommendation might 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, *General Key Management Guidance*, classifies the different types of keys and other cryptographic information according to their uses, discusses cryptoperiods and recommends appropriate cryptoperiods for each key type, provides recommendations and requirements for other keying material, introduces assurance of domain-parameter and public-key validity, discusses the implications of the compromise of keying material, and provides guidance on cryptographic algorithm strength selection implementation and replacement.
6. Section 6, *Protection Requirements for Cryptographic Information*, 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.
7. Section 7, *Key State and Transitions*, identifies the states in which a cryptographic key may exist during its lifetime.
8. Section 8, *Key Management Phases and Functions*, identifies four phases and a multitude of functions involved in key management. This section is of particular interest to cryptographic module vendors and developers of cryptographic infrastructure services.
9. Section 9, *Accountability, Audit, and Survivability*, discusses three control principles that are used to protect the keying material identified in Section 5.1.
10. Section 10, *Key Management Specifications for Cryptographic Devices or Applications,* specifies the content and requirements for key management

[1] The toolkit consists of publications specifying algorithms and guidance for their use, rather than software code.

specifications. Topics covered include the communications environment, component requirements, keying material storage, access control, accounting, and compromise recovery.

Appendices A and B are provided to supplement the main text where a topic demands a more detailed treatment. Appendix C contains a list of appropriate references, and Appendix D contains a list of changes since the originally published version of this document.

2 Glossary of Terms and Acronyms

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

2.1 Glossary

Access control	Restricts access to resources to only privileged entities.
Accountability	A property that ensures that the actions of an entity may be traced uniquely to that entity.
Algorithm originator-usage period	The period of time during which a specific cryptographic algorithm may be used by originators to apply protection to data (e.g., encrypt or generate a digital signature).
Algorithm security lifetime	The estimated time period during which data protected by a specific cryptographic algorithm remains secure.
Approved	FIPS-**approved** and/or NIST-recommended. An algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, or 2) specified elsewhere and adopted by reference in a FIPS or NIST Recommendation.
Archive	1. To place information into long-term storage. 2. A location or media used for long-term storage.
Association	A relationship for a particular purpose. For example, a key is associated with the application or process for which it will be used.
Assurance of (private key) possession	Confidence that an entity possesses a private key and its associated keying material.
Assurance of validity	Confidence that a public key or domain parameter is arithmetically correct.
Asymmetric key algorithm	See Public-key cryptographic algorithm.
Authentication	A process that provides assurance of the source and integrity of information in communications sessions, messages, documents or stored data.
Authentication code	A keyed cryptographic checksum based on an **approved** security function (also known as a Message Authentication Code).

Authorization	Access privileges that are granted to an entity; conveying 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 during the cryptoperiod of the key, if necessary.
Certificate	See Public-key certificate.
Certification authority	The entity in a Public Key Infrastructure (PKI) that issues certificates to certificate subjects.
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.
Contingency planning	The development of a contingency plan.
Cryptanalysis	1. Operations performed to defeat 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 in the algorithm itself.
Cryptographic algorithm	A well-defined computational procedure that takes variable inputs, including a cryptographic key, and produces an output.
Cryptographic boundary	An explicitly defined continuous perimeter that establishes the physical bounds of a cryptographic module and contains all hardware, software, and/or firmware components of a cryptographic module.
Cryptographic hash function	See Hash function.

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, reverse or verify 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 on data, 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 security properties (e.g., randomness) 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.
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-encryption key	A key used to encrypt and decrypt information other than keys.
Data integrity	A property whereby data has not been altered in an unauthorized manner since it was created, transmitted or stored.
Decryption	The process of changing ciphertext into plaintext using a cryptographic algorithm and key.
Deterministic random bit generator (DRBG)	A random bit generator that includes a DRBG algorithm and (at least initially) has access to a source of randomness. The DRBG produces a sequence of bits from a secret initial value called a seed, along with other possible inputs. A cryptographic DRBG has the additional property that the output is unpredictable, given that the seed is not known. A DRBG is sometimes also called a Pseudo Random Number Generator (PRNG) or a deterministic random number generator.

Digital signature	The result of a cryptographic transformation of data that, when properly implemented with a supporting infrastructure and policy, provides the services of: 1. Origin (i.e., source) authentication, 2. Data integrity authentication, and 3. Support for signer non-repudiation.
Distribution	See Key distribution.
Domain parameter	A parameter used in conjunction with some public-key algorithms to generate key pairs, to create digital signatures, or to establish keying material.
Encrypted key	A cryptographic key that has been encrypted using an **approved** security function 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 key	A cryptographic key that is generated for each execution of a key-establishment process and that meets other requirements of the key type (e.g., unique to each message or session). In some cases, ephemeral keys are used more than once within a single session (e.g., for broadcast applications) where the sender generates only one ephemeral key pair per message, and the private key is combined separately with each recipient's public key.
Hash-based message authentication code (HMAC)	A message authentication code that uses an **approved** keyed-hash function (i.e., [FIPS 198]).
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 resistant) 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.
Identifier	A bit string that is associated with a person, device or organization. It may be an identifying name, or may be something more abstract (for example, a string consisting of an IP address and timestamp), depending on the application.
Identity	The distinguishing character or personality of an entity.

Initialization vector (IV)	A vector used in defining the starting point of a cryptographic process.
Integrity (also, Assurance of integrity)	See Data integrity.
Integrity authentication	The process of providing assurance that data has not been modified since an authentication code was created for that data.
Integrity protection	See Integrity authentication.
Key	See Cryptographic key.
Key agreement	A key-establishment procedure where 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 independently of any other party's contribution.
Key component	See Cryptographic key component.
Key confirmation	A procedure used to provide assurance to one party that another party actually possesses the same keying material and/or shared secret.
Key de-registration	A function in the lifecycle of keying material; the marking of all keying material records and associations to indicate that the key is no longer in use.
Key derivation	The process by which one or more keys are derived from either a pre-shared key, or a shared secret (from a key-agreement scheme) and other information.
Key-derivation function	A function that, with the input of a cryptographic key or shared secret, and possibly other data, generates a binary string, called keying material.
Key-derivation key	A key used with a key-derivation function or method to derive additional keys. Sometimes called a master key.
Key-derivation method	A key-derivation function or other **approved** procedure for deriving keying material.
Key destruction	To remove all traces of keying material so that it cannot be recovered by either physical or electronic means.
Key distribution	The transport of a key and other keying material from an entity that either owns 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 to provide confidentiality protection. Also see Key-wrapping key.

Key establishment	A function in the lifecycle of keying material; the process by which cryptographic keys are securely established 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.
Key length	The length of a key in bits; used interchangeably with "Key size".
Key management	The activities involving the handling of cryptographic keys and other related security parameters (e.g., passwords) during the entire lifecycle of the keys, including their generation, storage, establishment, entry and output, use and destruction.
Key Management Policy	A high-level statement of organizational key management policies that identifies a high-level structure, responsibilities, governing standards, organizational dependencies and other relationships, and security policies.
Key Management Practices Statement	A document or set of documents 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 function in the lifecycle of keying material; mechanisms and processes that allow authorized entities to retrieve or reconstruct keying material from key backup or archive.
Key registration	A function in the lifecycle of keying material; the process of officially recording the keying material by a registration authority.
Key revocation	A function 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 size	The length of a key in bits; used interchangeably with "Key length".
Key transport	A key-establishment procedure whereby one party (the sender) selects and encrypts (or wraps) the keying material and then distributes the material to another party (the receiver). When used in conjunction with a public-key (asymmetric) algorithm, the keying material is encrypted using the public key of the receiver and subsequently decrypted using the private key of the receiver. When used in conjunction with a symmetric algorithm, the keying material is encrypted with a key-wrapping key shared by the two parties.
Key update	A function performed on a cryptographic key in order to compute a new key that is related to the old key.

Key-usage period	For a symmetric key, either the originator-usage period or the recipient-usage period.
Key wrapping	A method of cryptographically protecting keys using a symmetric key that provides both confidentiality and integrity protection.
Key-wrapping key	A symmetric key-encrypting key that is used to provide both confidentiality and integrity protection. Also see Key-encrypting key.
Keying material	The data (e.g., keys and IVs) necessary to establish and maintain cryptographic keying relationships.
Manual key transport	A non-automated means of transporting cryptographic keys by physically moving a device or document containing the key or key component.
Master key	See Key-derivation key.
Message authentication code (MAC)	A cryptographic checksum on data that uses an **approved** security function and a symmetric key to detect both accidental and intentional modifications of data.
Metadata	Information used to describe specific characteristics, constraints, acceptable uses and parameters of another data item (e.g., a cryptographic key).
NIST standards	Federal Information Processing Standards (FIPS) and NIST Recommendations.
Non-repudiation	A service using a digital signature that is used to support a determination of whether a message was actually signed by a given entity.
Operational phase (Operational use)	A phase in the lifecycle of keying material whereby keying material is used for standard cryptographic purposes.
Operational storage	The normal storage of operational keying material during its cryptoperiod.
Owner	For a static key pair, the entity that is associated with the public key and authorized to use the private key. For an ephemeral key pair, the owner is the entity that generated the public/private key pair. For a symmetric key, the owner is any entity that is authorized to use the key.
Originator-usage period	The period of time in the cryptoperiod of a 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, to verify access authorization or to derive cryptographic keys.
Period of protection	The period of time during which the integrity and/or confidentiality of a key needs to be maintained.

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 has a corresponding public key. Depending on the algorithm, the private key may be used, for example, to: 1. Compute the corresponding public key, 2. Compute a digital signature that may be verified by the corresponding public key, 3. Decrypt keys that were encrypted by the corresponding public key, or 4. Compute a shared secret during a key-agreement transaction.
Proof of possession (POP)	A verification process whereby assurance is obtained that the owner of a key pair actually has the private key associated with the public key.
Pseudorandom number generator (PRNG)	See Deterministic random bit generator (DRBG).
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 has a corresponding private key. The public key may be known by anyone and, depending on the algorithm, may be used, for example, to: 1. Verify a digital signature that is signed by the corresponding private key, 2. Encrypt keys that can be decrypted using the corresponding private key, or 3. Compute a shared secret during a key-agreement transaction.
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.

Random bit generator (RBG)	A device or algorithm that outputs a sequence of bits that appears to be statistically independent and unbiased. Also, see Random number generator.
Random number generator (RNG)	A process used to generate an unpredictable series of numbers. Also called a Random bit generator (RBG).
Recipient-usage period	The period of time during the cryptoperiod of a key in which the protected information is processed (e.g., decrypted).
Registration authority	A trusted entity that establishes and vouches for the identity of a user.
Retention period	The minimum amount of time that a key or other cryptographically related information should be retained in the archive.
RBG seed	A string of bits that is used to initialize a DRBG. Also just called a "seed."
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 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.
Secure communication protocol	A communication protocol that provides the appropriate confidentiality, source authentication, and data 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 life of data	The time period during which the security of the data needs to be protected (e.g., its confidentiality, integrity or availability).
Security services	Mechanisms used to provide confidentiality, integrity authentication, source authentication and/or support non-repudiation of information.
Security strength (Also “bits of security”)	A number associated with the amount of work (that is, the number of operations) that is required to break a cryptographic algorithm or system. In this Recommendation, the security strength is specified in bits and is a specific value from the set {80, 112, 128, 192, 256}. Note that a security strength of 80 bits is not longer considered sufficiently secure.
Seed	A secret value that is used to initialize a process (e.g., a DRBG). Also see RBG seed.
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 the 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 secret value that has been computed using a key-agreement scheme and is used as input to a key-derivation function/method.
Should	This term is used to indicate a very important recommendation. Ignoring the recommendation could result in undesirable results. Note that **should** may be coupled with **not** to become **should not**.
Signature generation	The use of a digital signature algorithm and a private key to generate a digital signature on data.
Signature verification	The use of a digital signature algorithm and a public key to verify a digital signature on data.
Source authentication	The process of providing assurance about the source of information. Sometimes called identity authentication or origin authentication.
Split knowledge	A process by which a cryptographic key is split into *n* multiple key components, each of which provides no knowledge of the original key. The components can be subsequently combined to recreate the original cryptographic key. If knowledge of *k* (where *k* is less than or equal to *n*) components is required to construct the original key, then knowledge of any *k*-1 key components provides no information about the original key other than, possibly, its length. Note that in this Recommendation, split knowledge is not intended to cover key shares, such as those used in threshold or multi-party signatures.
Static key	A key that is intended for use for a relatively long period of time and is typically intended for use in many instances of a cryptographic key-establishment scheme. Contrast with an Ephemeral key.
Symmetric key	A single cryptographic key that is used with a secret (symmetric) key algorithm.
Symmetric-key algorithm	A cryptographic algorithm that uses the same secret key for an operation and its complement (e.g., encryption and decryption).
System initialization	A function in the lifecycle of keying material; setting up and configuring a system for secure operation.

Trust anchor	1. An authoritative entity for which trust is assumed. In a PKI, a trust anchor is a certification authority, which is represented by a certificate that is used to verify the signature on a certificate issued by that trust-anchor. The security of the validation process depends upon the authenticity and integrity of the trust anchor's certificate. Trust anchor certificates are often distributed as self-signed certificates. 2. The self-signed public key certificate of a trusted CA.
Unauthorized disclosure	An event involving the exposure of information to entities not authorized access to the information.
User	See Entity.
User initialization	A function 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 function in the lifecycle of keying material; a process whereby an entity becomes a member of a security domain.
X.509 certificate	The X.509 public-key certificate or the X.509 attribute certificate, as defined by the ISO/ITU-T X.509 standard. Most commonly (including in this document), an X.509 certificate refers to the X.509 public-key certificate.
X.509 public-key certificate	A digital certificate containing a public key for an entity and a name for that entity, together with some other information that is rendered un-forgeable 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 Recommendation:

2TDEA	Two-key Triple Data Encryption Algorithm specified in [SP800-67].
3TDEA	Three-key Triple Data Encryption Algorithm specified in [SP800-67].
AES	Advanced Encryption Standard specified in [FIPS197].
ANS	American National Standard.
ANSI	American National Standards Institute.
CA	Certification Authority.
CRC	Cyclic Redundancy Check.
CRL	Certificate Revocation List.
DRBG	Deterministic Random Bit Generator.
DSA	Digital Signature Algorithm specified in [FIPS186].

ECC	Elliptic Curve Cryptography.
ECDSA	Elliptic Curve Digital Signature Algorithm specified in [ANSX9.62] and **approved** in [FIPS186].
FFC	Finite Field Cryptography.
FIPS	Federal Information Processing Standard.
HMAC	Keyed-Hash Message Authentication Code specified in [FIPS198].
IFC	Integer Factorization Cryptography.
IV	Initialization Vector.
MAC	Message Authentication Code.
NIST	National Institute of Standards and Technology.
PKI	Public-Key Infrastructure.
POP	Proof of Possession.
RA	Registration Authority.
RBG	Random Bit Generator.
RNG	Random Number Generator.
RSA	Rivest, Shamir, Adelman; an algorithm **approved** in [FIPS186] for digital signatures and in [SP800-56B] for key establishment.
SMIME	Secure Multipurpose Internet Mail Extensions.
TDEA	Triple Data Encryption Algorithm; Triple DEA specified in [SP800-67].
TLS	Transport Layer Security

3 Security Services

Cryptography may be used to perform or support several basic security services: confidentiality, integrity authentication, source 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 mechanism may provide more than one service (e.g., the use of digital signatures can provide integrity authentication, and source authentication), but not all services.

3.1 Confidentiality

Confidentiality is the property whereby information is not disclosed to unauthorized parties. Secrecy is a term that is often used synonymously with confidentiality. Confidentiality is achieved using encryption to render the information unintelligible except by authorized entities. The information may become intelligible again by using decryption. In order for encryption to provide confidentiality, the cryptographic algorithm and mode of operation must be designed and implemented so that an unauthorized party cannot determine the secret or private keys associated with the encryption or be able to derive the plaintext directly without deriving any keys.

3.2 Data Integrity

Data integrity is a property whereby data has not been altered in an unauthorized manner since it was created, transmitted or stored. Alteration includes the insertion, deletion and substitution of data. 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. 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.

In this Recommendation, the statement that a cryptographic algorithm "provides data integrity" means that the algorithm is used to detect unauthorized alterations. Authenticating integrity is discussed in the next section.

3.3 Authentication

Two types of authentication services can be provided using cryptography: integrity authentication and source authentication.

- An integrity authentication service is used to verify that data has not been modified, i.e., this service provides integrity protection.
- A source authentication service is used to verify the identity of the user or system that created information (e.g., a transaction or message).

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 an authentication service.

When multiple individuals are permitted to share the same source authentication information (such as a password or cryptographic key), it is sometimes called role-based authentication. See [FIPS140].

3.4 Authorization

Authorization is concerned with providing an official sanction or permission to perform a security function or activity (e.g., accessing a room). Authorization is considered as a security service that is often supported by a cryptographic service. Normally, authorization is granted after the execution of a successful source authentication[2] service. A non-cryptographic analog of the interaction between source authentication and authorization is the examination of an individual’s credentials to establish their identity (the source authentication process); after verifying the individual's identity and verifying that the individual is authorized access to some resource, such as a locked room, the individual is then provided with the key or password that will allow access to that room.

[2] Sometimes referred to as identity authentication.

Source authentication can also be used to authorize a role (such as a system administrator or audit role), rather than to identify an individual. Once authenticated for a role, an entity is authorized for all the privileges associated with that role.

3.5 Non-repudiation

In key management, non-repudiation is a term associated with digital signature keys and digital certificates that bind the name of the certificate subject to a public key. When non-repudiation is indicated for a digital signature key, it means that the signatures created by that key support not only the usual integrity and source authentication services of digital signatures, but also may (depending upon the context of the signature) indicate commitment by the certificate subject, in the same sense that a handwritten signature on a document may indicate commitment to a contract.

Digital signature keys in public key certificates may be designated in the certificate as digital signature or non-repudiation keys, or both. In practice, if non-repudiation is designated, a digital signature will normally also be designated.

A key for which only a digital signature is indicated (and non-repudiation is not indicated) is meant for source authentication, typically in a protocol such as TLS, where a certificate subject authenticates its identity by digitally signing a challenge with the private key. A key where only a digital signature is designated might also be used to sign an e-mail message to authenticate the source of that message. Regardless of the digital signature or non-repudiation designation, the digital signature can also be used to provide integrity authentication, as well.

If both digital signature and non-repudiation are indicated, that means that the key may be used not only to authenticate the source and provide integrity protection, but also, possibly, for "commitment," in the sense of accepting or agreeing to some terms or conditions. Whether or not commitment is implied, when a non-repudiation key is used to sign a message, its intent is determined by the message contents and the circumstances surrounding the signature. This is similar to the determination of whether a handwritten signature is simply an acknowledgement of receipt, or an agreement to some terms or conditions.

Where non-repudiation is indicated, certificate policies commonly include provisions intended to ensure that only one copy of the private key exists, and no party, other than the certificate subject, ever has control of that private key. This is done to protect against repudiation of the signature on the grounds that some party other than the certificate subject might have executed the signature.

In reality, a determination of non-repudiation is a legal decision with many aspects to be considered. Cryptographic mechanisms can only be used as one element in this decision.

3.6 Support Services

These basic cryptographic security services often require other 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, integrity authentication, source authentication, and support for 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 source 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 source authentication, as well as integrity authentication if the symmetric keys are unique to each pair of users.

2. A digital signature algorithm (Section 4.2.4) can provide source authentication and integrity authentication, as well as to support a non-repudiation decision.

3. Certain modes of encryption can provide confidentiality, integrity authentication, and source authentication when properly implemented. These modes **should** be specifically designed to provide these services.

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

Example:

Consider a system where 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 example system decide that a Public Key Infrastructure (PKI) needs to be established and that each entity wishing to communicate securely is required to physically prove his or her identity to a Registration Authority (RA). This identity-proving process requires the presentation of proper credentials, such as a driver’s license, passport or birth certificate. After establishing their correct identity, the individuals then generate a public static key pair in a smart card that is later used for key agreement. The public static key-agreement key is transferred from the smart card to the RA, where it is incorporated with the user identifier and other information into a digitally signed message for transmission to a Certification Authority (CA). The CA then composes the user’s public-key certificate by signing the public key of the user and the user’s identifier, 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 source-authentication and key-agreement purposes.

In this example, any two entities wishing to communicate may exchange public-key certificates containing public keys that are checked by verifying the CA’s signature on the certificate (using the CA’s public key). The public static key-agreement key of each of the two entities and each entity's own private static key-agreement key are then used in a key-agreement scheme to produce a shared secret that is known by 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 are used, e.g., one of the shared keys is used to encrypt for confidentiality, while another key is used for data integrity and source authentication. The receiver of the data protected by the key(s) has assurance that the data came from the other entity indicated by the public-key certificate, that the data remains confidential, and that the integrity of the data is preserved.

Alternatively, if confidentiality is not required, integrity authentication and source authentication can be attained by establishing a digital-signature key pair and corresponding certificate for each entity. The private signature key of the sender is used to sign the data, and the sender's public signature-verification key is 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 the identity authentication process,

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

e. The care taken by the users in generating their keys and protecting them from unauthorized use.

Therefore, the design of a security system that provides the 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 Recommendation advises the users of cryptographic mechanisms on the appropriate choices of algorithms and key sizes.

This section describes the **approved** cryptographic algorithms that provide security services, such as confidentiality, integrity authentication, and source authentication.

4.1 Classes of Cryptographic Algorithms

There are three basic classes of **approved** cryptographic algorithms: hash functions, 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 functions do not require keys for their basic operation. Hash functions generate a relatively small digest (hash value) from a (possibly) large input in a way that is fundamentally difficult to reverse (i.e., it is hard to find an input that will produce a given output). Hash functions are used as building blocks for key management, for example,

1. To provide source and integrity authentication services (Section 4.2.3) – the hash function is used with a key to generate a message authentication code;
2. To compress messages for digital signature generation and verification (Section 4.2.4);
3. To derive keys in key-establishment algorithms (Section 4.2.5); and
4. To generate deterministic random numbers (Section 4.2.7).

Symmetric-key algorithms (sometimes known as secret-key algorithms) transform data in a way that is fundamentally difficult to undo without knowledge of a secret key. The key is "symmetric" because the same key is used for a cryptographic operation and its inverse (e.g., encryption and decryption). Symmetric keys are often known by more than one entity; however, the key **shall not** be disclosed to entities that are not authorized access to the data protected by that algorithm and key. Symmetric key algorithms are used, for example,

1. To provide data confidentiality (Section 4.2.2); the same key is used to encrypt and decrypt data;
2. To provide source and integrity authentication 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); and
4. To generate deterministic random numbers (Section 4.2.7).

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[3]. Even though the public and private keys of a key pair are related, knowledge of the public key cannot be used to determine the private key. Asymmetric algorithms are used, for example,

1. To compute digital signatures (Section 4.2.4), and
2. To establish cryptographic keying material (Section 4.2.5)

[3] Sometimes a key pair is generated by a party that is trusted by the key owner.

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 Functions

Many algorithms and schemes that provide a security service use a hash function as a component of the algorithm. Hash functions can be found in digital signature algorithms (see [FIPS186]), Keyed-Hash Message Authentication Codes (HMAC) (see [FIPS198]), key-derivation functions/methods (see [SP800-56A], [SP800-56B], [SP800-56C] and [SP800-108]), and random number generators (see [SP800-90]). **Approved** hash functions are defined in [FIPS180] and [FIPS202].

A hash function takes an input of arbitrary length and outputs a fixed-length 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 cryptographic hash function, it is not feasible to find a message that will produce a given hash value (pre-image resistance), nor is it feasible to find two messages that produce the same hash value (collision resistance).

Several hash functions are **approved** for Federal Government use and are defined in [FIPS180] and FIPS 202. Algorithm standards need to specify either the appropriate size for the hash function or provide the hash-function selection criteria if the algorithm can be configured to use different hash functions.

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 TDEA. Each of these algorithms operates on blocks (chunks) of data during an encryption or decryption operation. For this reason, these algorithms are commonly called block cipher algorithms.

4.2.2.1 Advanced Encryption Standard (AES)

The AES algorithm is specified in [FIPS197]. AES encrypts and decrypts data in 128-bit blocks, using 128, 192 or 256-bit keys. The nomenclature for AES for the different key sizes is AES-*x*, where *x* is the key size (e.g., AES-256). All three key sizes are considered adequate for most Federal Government applications.

4.2.2.2 Triple DEA (TDEA)

Triple DEA is defined in [SP800-67]. TDEA encrypts and decrypts data in 64-bit blocks, using three 56-bit keys. Two variations of TDEA have been defined: two-key TDEA (2TDEA), in which the first and third keys are identical, and three-key TDEA, in which the three keys are all different (i.e., distinct).

The use of two-key TDEA will no longer be approved for applying cryptographic protection (e.g., encryption) after December 31, 2015 (see [SP800-131A]); however, two-key TDEA may continue to be used for processing already-protected information (e.g., decryption).

Federal applications **shall** only use three distinct keys whenever using TDEA for applying cryptographic protection after the end of 2015; see Table 2 in Section 5.6.1 and [SP800-131A] for further guidance.

4.2.2.3 Modes of Operation

With a block-cipher encryption operation, 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 are encrypted separately, an adversary can easily substitute individual blocks, possibly without detection. Furthermore, certain kinds of data patterns in the plaintext, such as repeated blocks, are apparent in the ciphertext.

Cryptographic modes of operation have been defined to alleviate this problem by combining the basic cryptographic algorithm with variable initialization vectors and some sort of feedback of the information derived from the cryptographic operation. The NIST Recommendation for Block Cipher Modes of Operation [SP800-38A] defines modes of operation for the encryption and decryption of data using block cipher algorithms, such as AES and TDEA. Other modes **approved** for encryption are specified in other parts of [SP800-38]; some of these modes also produce message authentication codes (see Section 4.2.3). Guidance on the secure use of each mode is provided for each mode in addition to the mode specification.

Note that one of the modes included in [SP800-38A] is the ECB mode. This mode is not recommended for general use, as the ciphertext leaks information about plaintext after relatively small amounts of data are encrypted.

4.2.3 Message Authentication Codes (MACs)

Message Authentication Codes (MACs) can be used to provide source and integrity authentication. A MAC is a cryptographic checksum on the data that is used in order to provide assurance that the data has not changed and that the MAC was computed by the expected entity. Although message (i.e., data) 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 that the originator (i.e., the source) of the data is a key holder (i.e., an entity authorized to have the key). MACs are often used to authenticate the originator to the recipient when only those two parties share the MAC key.

The computation of a MAC requires the use of (1) a secret key that is known only by the party that generates the MAC and by the intended recipient(s) of the MAC, and (2) the data on which the MAC is calculated. The result of the MAC computation is often called a MacTag when transmitted; a MacTag is either a full-length or truncated result from the MAC computation. 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

[SP800-38B] defines a mode to compute a MAC using **approved** block cipher algorithms, such as AES and TDEA. The key and block size used to compute the MAC depend on the algorithm used. If the same block cipher is used for both encryption and MAC computation in two separate cryptographic operations (i.e., using an encryption mode from [SP800-38A] and a MAC computed as specified in [SP800-38B]), then the same key **shall not** be used for both the

MAC and encryption operations. Note that some other modes of operation specified in [SP800-38] perform encryption, integrity authentication and source authentication[4] using a single key.

4.2.3.2 MACs Using Hash Functions

[FIPS198] 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, which is the MAC algorithm specified in [FIPS198]; the choice of key size depends on the amount of security to be provided to the data and the hash function used. See [SP800-107] for further discussions about HMAC, and Section 5.6 of this Recommendation (i.e., SP 800-57, Part 1) for further discussion.

4.2.4 Digital Signature Algorithms

Digital signatures are used to provide source authentication, integrity authentication and support non-repudiation. Digital signatures are used in conjunction with hash functions and are computed on data of any length (up to a limit that is determined by the hash function). [FIPS186] specifies algorithms that are **approved** for the computation of digital signatures[5]. It defines the Digital Signature Algorithm (DSA) and adopts the RSA algorithm, as specified in [ANSX9.31] and [PKCS#1] (version 1.5 and higher), and the ECDSA algorithm, as specified in [ANSX9.62].

[FIPS186] also specifies several **approved** key sizes for each of these algorithms, and includes methods for generating the algorithm's key pairs and any other parameters needed for digital signature generation and verification. Note that older systems (legacy systems) used smaller key sizes than those currently provided in [FIPS186]. Digital signature generation **shall** be performed using keys that meet or exceed the key sizes specified in [FIPS186] and using key pairs that are generated in accordance with [FIPS186]. Smaller key sizes **shall only** be used to verify signatures that were generated using those smaller keys. See [SP800-131A].

4.2.5 Key Establishment Schemes

Automated key-establishment schemes are used to set up keys to be used between communicating entities. Two types of automated key-establishment schemes are defined: key transport and key agreement. **Approved** key-establishment schemes are provided in [SP800-56A] and [SP800-56B].

Key transport is the distribution of a key (and other keying material) from one entity (the sender) to another entity (the receiver). The keying material is encrypted by the sending entity and decrypted by the receiving entity(ies). If a symmetric algorithm (e.g., AES) is used to transport a key, the algorithm is used to wrap (i.e., encrypt) the keying material to be distributed; the sending and receiving entities need to know the symmetric key-wrapping key (i.e., the key-encrypting key). See Section 4.2.5.4 for further discussion on key encryption and key wrapping.

---

[4] See the caveat regarding source authentication in Section 4.2.3 above.

[5] Two general types of digital signature methods are discussed in literature: digital signatures with appendix, and digital signatures with message recovery. [FIPS186] specifies algorithms for digital signatures with appendix, and is the digital signature method that is discussed in this Recommendation.

If a public-key algorithm is used for key transport, , one key of a key pair is used to encrypt the key to be established, and the other key is used for decryption. In this case, the sending entity encrypts the keying material using the receiving entity's public key, and the receiving entity decrypts the received keying material using the associated private key.

Key agreement is the participation by both 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-key algorithm is used, each entity shares the same symmetric key-wrapping key.

4.2.5.1 Discrete Log Key Agreement Schemes

[SP800-56A] specifies key-establishment schemes that use discrete-logarithm-based public-key algorithms. These schemes are specified using either finite-field math (the form of math that most of us use) or elliptic curve math.

With the key-establishment schemes specified in [SP800-56A], a party may own and use an ephemeral key, a static key, or both an ephemeral and a static key in a single key-agreement transaction. The ephemeral key is used to provide a new secret for each key-establishment transaction, while the static key (if used in a PKI with public-key certificates) provides for the authentication of the owner.

[SP800-56A] also provides a key-confirmation method for most of its schemes to obtain assurance that each party has agreed upon the same keying material (see Section 4.2.5.5 for a discussion of key confirmation).

4.2.5.2 Key Establishment Using Integer-Factorization Schemes

[SP800-56B] provides key-establishment schemes that use integer-factorization-based public-key algorithms (e.g., RSA). Two of the families of schemes specified in [SP800-56B] provide for key agreement, and the other two families provide for key transport. Each scheme family has a basic scheme and one or more schemes that provide key confirmation.

In these schemes, one party always owns and uses a key pair, and the other party may or may not use a key pair, depending on the scheme. Only static keys are used in the [SP800-56B] schemes; ephemeral keys are not used.

4.2.5.3 Security Properties of the Key-Establishment Schemes

Cryptographic protocol designers need to understand the security properties of the schemes in order to assure that the desired capabilities are available to the user. In general, schemes where each party uses both an ephemeral and a static key provide more security properties than schemes using fewer keys. However, it may not be practical for both parties to use both static and ephemeral keys in certain applications, and the use of ephemeral keys is not specified for all algorithms (see [SP800-56B]). For example, in email applications, it is desirable to send messages to other parties who are not on-line. In this case, the receiver cannot be expected to provide an ephemeral key to establish the message-encrypting key during a [SP800-56A] key-agreement scheme.

Both [SP80056A] and [SP800-56B] include discussions of the security properties of each of its schemes.

4.2.5.4 Key Encryption and Key Wrapping

Key encryption provides confidentiality protection for a key by encrypting that key using a key-encrypting key; decryption reverses the process using the same key. Key wrapping provides both confidentiality and integrity protection for a key using a key-wrapping key to both encrypt and integrity protect the key to be protected; key unwrapping decrypts the ciphertext key and verifies its integrity. Although the key-protection services are slightly different and use different methods, the keys are generated in the same manner. In this Recommendation and elsewhere, the terms[6] are often used interchangeably.

Both processes use a symmetric algorithm, such as AES. Several methods for key wrapping have been specified or referenced in [SP800-38F].

4.2.5.5 Key Confirmation

Key confirmation is used by two parties in a key-establishment process to provide assurance that common keying material and/or a shared secret[7] has been established. The assurance may be provided to only one party (unilateral) or it may be provided to both parties (bilateral). The assurance may be provided as part of the key-establishment scheme, or it may be provided by some action that takes place outside of the scheme. For example, after a key is established, two parties may provide assurance (i.e., a confirmation) to one another that they possess the same key by demonstrating their ability to encrypt and decrypt data intended for each other.

[SP800-56A] provides for unilateral key confirmation for schemes where one party has a static key-establishment key, and bilateral key confirmation for schemes where both parties have static key-establishment keys. A total of ten key-confirmation schemes are provided, seven of which are unilateral, and three of which are bilateral.

[SP800-56B] provides for unilateral key confirmation from the responder, in the case of a key agreement scheme, and from the receiver, in the case of a key-transport scheme. Initiator and bilateral key confirmation are also provided for one family of key-agreement schemes.

4.2.6 Key Establishment Protocols

Key establishment protocols use key-establishment schemes in order to specify the processing necessary to establish a key. However, key-establishment protocols also specify message flow and format. Key-establishment protocols need to be carefully designed to not give secret information to a potential attacker. For example, a protocol that indicates abnormal conditions, such as an integrity error, may permit an attacker to confirm or reject an assumption regarding secret data. Alternatively, if the time or power required to perform certain computations are based upon the value of the secret or private key in use, then an attacker may be able to deduce the key from observed fluctuations.

Therefore, it is best to design key-establishment protocols so that:

1. The protocols do not provide for an early exit from the protocol upon detection of a single error,

---

[6] I.e., key-encrypting key and key-wrapping key, encrypt and wrap, and decrypt and unwrap.

[7] An intermediate value computed during a key-agreement scheme.

2. The protocols trigger an alarm after a certain reasonable number of detected error conditions, and
3. The key-dependent computations are obscured from the observer in order to prevent or minimize the detection of key-dependent characteristics.

4.2.7 Random Bit Generation

Random bit generators (RBGs) (also called random number generators (RNGs)) are required for the generation of keying material (e.g., keys and IVs). RBGs generate sequences of random bits (e.g., 010011); technically, RNGs translate those bits into numbers (e.g., 010011 is translated into the number 19). However, the use of the term "random number generator" (RNG) is commonly used to refer to both concepts

Two classes of RBGs are defined: deterministic and non-deterministic. Deterministic Random Bit Generators (DRBGs), sometimes called deterministic random number generators or pseudorandom number generators, use cryptographic algorithms and the associated keying material to generate pseudorandom bits from an initial value, called a seed, that provides entropy (i.e., randomness) to the process. Depending on the implemented DRBG design or the environment, additional entropy never be introduced again, although such additional entropy is recommended. [SP800-90A] specifies DRBG algorithms that may be used to generate random bits for cryptographic applications (e.g., key or IV generation).

Non-deterministic Random Bit Generators (NRBGs), sometimes called true RNGs, use some unpredictable physical source that is outside human control to introduce new entropy for every bit output by the NRBG. The unpredictable source is commonly known as an entropy source. [SP800-90B] provides guidance on the implementation and testing of entropy sources.

[SP800-90C] has been developed to provide guidance on the construction of DRBGs and NRBGs from the algorithms in [SP800-90A] and entropy sources that comply with [SP800-90B].

5 GENERAL KEY MANAGEMENT GUIDANCE

This section classifies the different types of keys and other cryptographic information according to their uses; discusses cryptoperiods and recommends appropriate cryptoperiods for each key type; provides recommendations and requirements for other keying material; introduces assurance of domain-parameter validity, public-key validity, and private-key possession; discusses the implications of the compromise of keying material; and provides guidance on the selection, implementation, and replacement of cryptographic algorithms and key sizes according to their security strengths.

5. 1 Key Types and Other Information

There are several different types of cryptographic keys, each used for a different purpose. In addition, there is other information that is specifically related to cryptographic algorithms and keys.

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. See Table 5 in Section 6.1.1 for the required protections for each type of information.

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 source authentication, integrity authentication and support the non-repudiation of messages, documents or stored data.

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 that are intended to provide source authentication, integrity authentication and support the non-repudiation of messages, documents or stored data.

3. *Symmetric authentication key*: Symmetric authentication keys are used with symmetric-key algorithms to provide source authentication and assurance of the integrity of communication sessions, messages, documents or stored data (i.e., integrity authentication).

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 of the identity of an originating entity (i.e., the source) when establishing an authenticated communication session[8].

5. *Public authentication key*: A public authentication key is the public key of an asymmetric (public) key pair that is used with a public-key algorithm to provide assurance of the identity of an originating entity (i.e., the source) when establishing an authenticated communication session[9].

6. *Symmetric data-encryption key*: These keys are used with symmetric-key algorithms to apply confidentiality protection to information (i.e., to encrypt the information). The same key is also used to remove the confidentiality protection (i.e., to decrypt the information).

7. *Symmetric key-wrapping key*: Symmetric key-wrapping keys (also called key-encrypting keys) are used to encrypt other keys using symmetric-key algorithms. The key-wrapping key used to encrypt a key is also used to reverse the encryption operation (i.e., to decrypt the encrypted key). Depending on the algorithm with which the key is used, the key may also be used to provide integrity protection.

8. *Symmetric random number generation keys*: These keys are used to generate random numbers or random bits.

9. *Symmetric master key*: A symmetric master key is used to derive other symmetric keys (e.g., data-encryption keys, key-wrapping keys, or source authentication keys) using symmetric cryptographic methods. The master key is also known as a key-derivation key.

[8] While integrity protection is also provided, it is not the primary intention of this key.

[9] While integrity protection is also provided, it is not the primary intention of this key.

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 corresponding 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). The encrypted form of the established key might be stored for later decryption using the private key-transport key.

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 long-term 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 long-term public 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).

15. *Private ephemeral key-agreement key*: Private ephemeral key-agreement keys are the short-term private keys of asymmetric (public) key pairs that are used only once[10] 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 short-term public keys of asymmetric key pairs that are used in a single key-establishment transaction[11] 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 entity responsible for monitoring and granting access privileges for authorized entities and by 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.

[10] In some cases ephemeral keys are used more than once, though within a single "session". For example, when Diffie-Hellman is used in S/MIME CMS, the sender may generate one ephemeral key pair per message, and combine the private key separately with each recipient's public key.

[11] The public ephemeral key-agreement key of a sender may be retained by the receiver for later use in decrypting a stored (encrypted) message for which the ephemeral key pair was generated.

19. *Public authorization key*: A public authorization key is the public key of an asymmetric (public) key pair that is used to verify privileges for an entity that knows the associated private authorization key.

5.1.2 Other Cryptographic or Related Information

Other information used in conjunction with cryptographic algorithms and keys also needs to be protected. See Table 6 in Section 6.1.2 for the required protections for each type of information.

1. *Domain Parameters*: Domain parameters are used in conjunction with some public-key algorithms to generate key pairs, to create digital signatures or to establish keying material.
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)
3. *Shared Secrets:* Shared secrets are generated during a key-agreement process as defined in [SP800-56A] and [SP800-56B]. Shared secrets **shall** be protected and handled in the same manner as cryptographic keys. If a FIPS 140-validated cryptographic module is being used, then the protection of the shared secrets is provided by the cryptographic module.
4. *RBG seeds*: RBG seeds are used in the generation of *deterministic random* bits (e.g., used to generate keying material that must remain secret or private).
5. *Other public information*: Public information (e.g., a nonce) is often used in the key-establishment process.
6. *Other secret information*: Secret information may be included in the seeding of an RBG or in the establishment of keying material.
7. *Intermediate Results*: The intermediate results of cryptographic operations using secret information must be protected. Intermediate results **shall not** be available for purposes other than as intended.
8. *Key-control information*: Information related to the keying material (e.g., the identifier, purpose, or a counter) must be protected to ensure that the associated keying material can be correctly used. The key-control information is included in the metadata associated with the key (see Section 6.2.3.1).
9. *Random numbers* (or bits): The random numbers created by a random bit generator **should** be protected when retained. When used directly as keying material or in its generation, the random bits **shall** be protected as discussed in Section 6.
10. *Passwords*: A password is used to acquire access to privileges and can be used as a credential in a source authentication mechanism. A password can also be used to derive cryptographic keys that are used to protect and access data in storage, as specified in [SP800-132].
11. *Audit information*: Audit information contains a record of key-management events.

5.2 Key Usage

In general, a single key **shall** be used for only one purpose (e.g., encryption, integrity authentication, key wrapping, random bit generation, or digital signatures). There are several reasons for this:

1. The use of the same key for two different cryptographic processes may weaken the security provided by one or both of the processes.

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. In this case, the private key is used as both a private key-transport key to decrypt the encrypted keys and as a private signature key to apply digital signatures. It may be necessary to retain the private key-transport key beyond the cryptoperiod of the corresponding public key-transport key in order to decrypt the encrypted keys needed to access encrypted data. On the other hand, the private signature key **shall** be destroyed at the expiration of its cryptoperiod to prevent its compromise (see Section 5.3.6). In this example, the longevity requirements for the private key-transport key and the private digital-signature key 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 integrity authentication and source authentication using a single digital signature, or when a single symmetric 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.

This Recommendation permits the use of a private key-transport or key-agreement key to generate a digital signature for the following special case:

When requesting the (initial) certificate for a static key-establishment key, the corresponding private key may be used to sign the certificate request. Also refer to Section 8.1.5.1.1.2.

5.3 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).

Sometimes cryptoperiods are defined by an arbitrary time period or maximum amount of data protected by the key. However, trade-offs associated with the determination of cryptoperiods involve the risk and consequences of exposure, which should be carefully considered when selecting the cryptoperiod (see Section 5.6.4).

5.3.1 Risk Factors Affecting Cryptoperiods

Among the factors affecting the length of a cryptoperiod are:

1. The strength of the cryptographic mechanisms (e.g., the algorithm, key length, block size, and mode of operation),
2. The embodiment of the mechanisms (e.g., a [FIPS140] Level 4 implementation or a software implementation on a personal computer),
3. The operating environment (e.g., a 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 derivation, or key protection),
7. The re-keying method (e.g., keyboard entry, re-keying using a key loading device where humans have no direct access to key information, or remote re-keying 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,
11. Personnel turnover (e.g., CA system personnel), and
12. The threat to the information from adversaries (e.g., whom the information is protected from, and what are their perceived technical capabilities and financial resources to mount an attack).
13. The threat to the information from new and disruptive technologies (e.g., quantum computers).

In general, short cryptoperiods enhance security. For example, some cryptographic algorithms might be less vulnerable to cryptanalysis if the adversary has only a limited amount of information encrypted under a single key. 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 key 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 **approved** algorithms, modes of operation, and key sizes, adversaries may be able to access keys through the penetration or subversion of a system with less expenditure of time and resources than would be required to mount and execute a cryptographic attack.

5.3.2 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 subject to the caveat regarding the security and integrity of the re-keying, key update or key-derivation process (see Sections 8.2.3 and 8.2.4). Short cryptoperiods may be counter productive, particularly where denial of service is the paramount concern, and there is a significant potential for error in the re-keying, key update or key-derivation process.

5.3.3 Other Factors Affecting Cryptoperiods

5.3.3.1 Communications versus Storage

Keys that are used for confidentiality protection of communication exchanges may often have shorter cryptoperiods than keys used for the protection of stored data. Cryptoperiods are generally made longer for stored data because the overhead of re-encryption associated with changing keys may be burdensome.

5.3.3.2 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 the security measures necessary to support longer cryptoperiods may be justified (e.g., costly and inconvenient physical, procedural, and logical access security; and the use of cryptography strong enough to support longer cryptoperiods, even where this may result in significant additional processing overhead). In other cases, the cryptoperiod may be shorter than would otherwise be necessary; for example, keys may be changed frequently in order to limit the period of time that the key management system maintains status information.

5.3.4 Asymmetric Key Usage Periods and Cryptoperiods

For key pairs, each key of the pair has its own cryptoperiod. One key of the key pair is used to apply cryptographic protection (e.g., create a digital signature), and its cryptoperiod can be considered as an "originator-usage period." The other key of the key pair is used to process the protected information (e.g., verify a digital signature); its cryptoperiod is considered to be the "recipient-usage period." The key pair's originator and recipient-usage periods typically begin at the same time, but the recipient-usage period may extend beyond the originator-usage period. For example:

- In the case of digital signature key pairs, the private signature key is used to sign data (i.e., apply cryptographic protection), so its cryptoperiod is considered to be an originator-usage period. The public signature-verification key is used to verify digital signatures (i.e., process already-protected information); its cryptoperiod is considered to be a recipient-usage period.

  For a private signature key that is used to generate digital signatures as a proof-of-origin (i.e., for source authentication), the originator-usage period (i.e., the period during which the private key may be used to generate signatures) is often shorter than the recipient-usage period (i.e., the period during which the signature may be verified). In this case, the private key is intended for use for a fixed period of time, after which time the key owner **shall** destroy[12] the private key. The public key may be available for a longer period of time for verifying signatures.

  The cryptoperiod of a private source-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 source-authentication key). That is, when the private key will not be used to sign challenges, the public key is no longer needed. In this case, the originator and recipient-usage periods are the same.

- For key transport keys, the public key-transport key is used to apply protection (i.e., encrypt), so its cryptoperiod would be considered as an originator-usage period; the private key-transport key is used to decrypt, so its cryptoperiod would be considered as the recipient-usage period. The originator-usage period (i.e., the period during which the public key may be used for encryption) is often shorter than the recipient-usage period (i.e., the period during which the encrypted information may be decrypted).

- For key-agreement algorithms, the cryptoperiods of the two keys of the key pair are usually the same.

Where public keys are distributed in public-key certificates, each certificate has a validity period, indicated by the *notBefore* and *notAfter* dates in the certificate. Certificates may be renewed, i.e., a new certificate containing the same public key may be issued with a new validity period. The sum of the validity periods for the original certificate and all renewed certificates for the same public key **shall not** exceed the cryptoperiod of the key of the key pair used to apply protection (i.e., the key with the originator-usage period).

See Section 5.3.6 for guidance regarding specific key types.

5.3.5 Symmetric Key Usage Periods and 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 period of time during which cryptographic protection may be applied to data is called the *originator-usage period*, and the period of time during

[12] A simple deletion of the keying material might not completely obliterate the information. For example, erasing the information might require overwriting that information multiple times with other non-related information, such as random bits, or all zero or one bits. Keys stored in memory for a long time can become "burned in". This can be mitigated by splitting the key into components that are frequently updated (see [DiCrescenzo]).

which the protected information is processed is called the *recipient-usage period*. A symmetric key **shall not** be used to provide protection after the end of the originator-usage period. The recipient-usage period may extend beyond the originator-usage period (see Figure 1). This permits all information that has been protected by the originator to be processed by the recipient before the processing key is deactivated. However, in many cases, the originator and recipient-usage periods are the same. The (total) "cryptoperiod" of a symmetric key is the period of time from the beginning of the originator-usage period to the end of the recipient-usage period, although the originator-usage period has historically been used as the cryptoperiod for the key.

Note that in some cases, predetermined cryptoperiods may not be adequate for the security life of the protected data. If the required security life exceeds the cryptoperiod, then the protection will need to be reapplied using a new key.

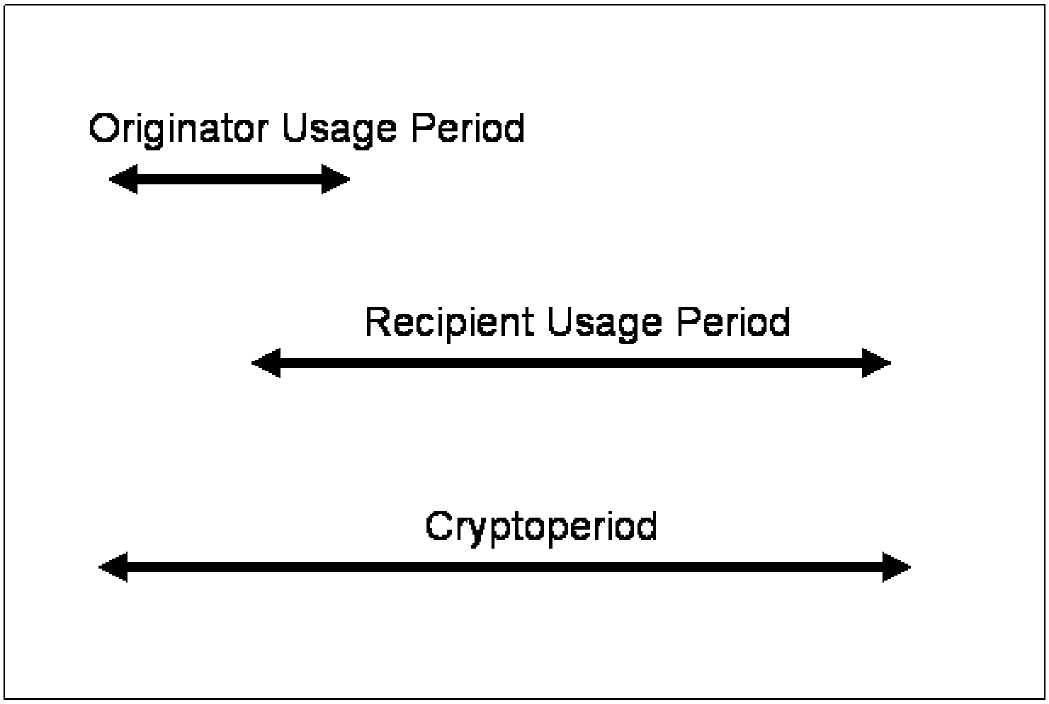

Figure 1: Symmetric key cryptoperiod

Examples of the use of the usage periods include:

a. When a symmetric key is used only for securing communications, the period of time from the originator's application of protection to the recipient's processing may be 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.

c. When a symmetric key is used to protect stored information, the recipient-usage period may start after the beginning of the originator-usage period as shown in Figure 1. For example, information may be encrypted before being stored on some storage media. At some later time, the key may be distributed in order to decrypt and recover the information.

5.3.6 Cryptoperiod Recommendations for Specific Key Types

The cryptoperiod required for a given key may be affected by the key type as much as by the usage environment and data characteristics described above. Some general cryptoperiod recommendations for various key types are suggested below. Note that the cryptoperiods suggested are only rough order-of-magnitude guidelines; longer or shorter cryptoperiods may be warranted, depending on the application and environment in which the keys will be used. However, when assigning a longer cryptoperiod than that suggested below, serious consideration should be given to the risks associated with doing so (see Section 5.3.1). Most of the suggested cryptoperiods are on the order of 1-2 years, based on 1) a desire for maximum operational efficiency and 2) assumptions regarding the minimum criteria for the usage environment (see [FIPS140], [SP800-14], and [SP800-37]). The factors described in Sections 5.3.1 through 5.3.3 **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. When the corresponding public key has been certified by a CA, the cryptoperiod ends when the *notAfter* date is reached on the last certificate issued for the public key[13].

   b. Cryptoperiod: Given the use of **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 one to three years is recommended. The key **shall** be destroyed at the end of its cryptoperiod.

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 corresponding private signature key needs to be verified. A longer cryptoperiod for the public signature-verification key (than the private signature key) poses a relatively minimal security concern.

[13] Multiple consecutive certificates may be issued for the same public key, presumably with different *notBefore* and *notAfter* validity dates.

b. Cryptoperiod: The cryptoperiod may be on the order of several years, though due to the long exposure of protection mechanisms to hostile attack, the 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, the consequences of exposure to attacks on physical, procedural, and logical access-control mechanisms for the private key are not affected.

Some systems use a cryptographic timestamping function to place an unforgeable timestamp on each signed message. Even though the cryptoperiod of the private signature key has expired, the corresponding public signature-verification key may be used to verify signatures on messages whose timestamps are within the cryptoperiod of the private signature key. In this case, one is relying on the cryptographic timestamp function to assure that the message was signed within the signature key's originator-usage period.

3. *Symmetric authentication key*:

a. Type Considerations: The cryptoperiod of a symmetric authentication key[14] depends on the sensitivity of the type of information it protects and the protection afforded by the key. For very sensitive information, the authentication key may need to be unique to the protected information. For less sensitive information, suitable cryptoperiods may extend beyond a single use of the key. The originator-usage period of a symmetric authentication key applies to the use of that key in applying the original cryptographic protection for the information (e.g., computing the MAC); new MACs **shall not** be computed on information using that key after the end of the originator-usage period. However, the key may need to be available to verify the MAC on the protected data beyond the originator-usage period (i.e., the recipient-usage period extends beyond the originator-usage period). The recipient-usage period is the period during which a MAC generated during the originator-usage period needs to be verified. Note that if a MAC key is compromised, it may be possible for an adversary to modify the data and then recalculate the MAC.

b. Cryptoperiod: Given the use of **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 originator-usage period of up to two years is recommended, and a maximum recipient-usage period of three years beyond the end of the originator-usage period is recommended.

4. *Private authentication key*:

a. Type Considerations: A private authentication key[15] may be used multiple times. Its corresponding public key could be certified, for example, by a Certification Authority. In most cases, the cryptoperiod of the private authentication key is the same as the cryptoperiod of the corresponding public key.

[14] Used to enable data integrity and source authentication.

[15] Which may be used to enable data integrity and source authentication, as well as non-repudiation.

b. Cryptoperiod: An appropriate cryptoperiod for a private authentication key would be one to two 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 corresponding private authentication key. The cryptoperiod is, in effect, the period during which the identity of the originator of information protected by the corresponding private authentication key needs to be verified, i.e., the information source needs to be authenticated[16].

b. Cryptoperiod: An appropriate cryptoperiod for the public authentication key would be one to two 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 link encryption) **should** have a relatively short originator-usage period. An encryption key used to encrypt less information over time could have a longer originator-usage period. 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 5.3.5).

During the originator-usage period, an encryption of the information may be performed using the data-encryption key; the key **shall not** be used for performing an encryption operation on information beyond this period. However, the key may need to be available to decrypt the protected data beyond the originator-usage period (i.e., the recipient-usage period may need to extend beyond the originator-usage period).

b. Cryptoperiod: The originator-usage period recommended for the encryption of large volumes of information over a short period of time (e.g., for link encryption) is on the order of a day or a week. An encryption key used to encrypt smaller volumes of information might have an originator-usage period of up to two years. A maximum recipient-usage period of three years beyond the end of the originator-usage period is recommended.

In the case of symmetric data-encryption keys that are used to encrypt single messages or single communications sessions, the lifetime of the protected data could be months or years because the encrypted messages may be stored for later reading. Where information is maintained in encrypted form, the symmetric data-encryption keys need to be maintained until that information is re-encrypted under a new key or destroyed. Note that confidence in the confidentiality of the information is reduced with the passage of time.

[16] While integrity protection is also provided, it is not the primary intention of this key.

7. *Symmetric key-wrapping key*:

a. Type Considerations: A symmetric key-wrapping key that is used to wrap (i.e., encrypt and integrity protect) very large numbers of keys over a short period of time **should** have a relatively short originator-usage period. If a small number of keys are wrapped, the originator-usage period 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 key-wrapping protection for the keys; a wrapping operation **shall not** be performed using a key-wrapping key whose originator-usage period has expired. However, the key-wrapping key may need to be available to unwrap the protected keys (i.e., decrypt and verify the integrity of the wrapped keys) beyond the originator-usage period (i.e., the recipient-usage period may need to extend beyond the originator-usage period); the recipient-usage period is the period of time during which keys wrapped during the key-wrapping key's originator-usage period may need to be unwrapped.

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 (i.e., which includes both the originator and recipient-usage periods) is a single communication session. It is assumed that the wrapped key will not be retained in its wrapped form, so the originator-usage period and recipient-usage period of the key-wrapping key is the same. In other cases, key-wrapping keys may be retained so that the files or messages encrypted by the wrapped keys may be recovered later on. In this case the recipient-usage period may be significantly longer than the originator-usage period of the key-wrapping key, and cryptoperiods lasting for years may be employed.

b. Cryptoperiod: The recommended originator-usage period for a symmetric key-wrapping key that is used to wrap 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 wrapped under the key-wrapping key, the originator-usage period of the key-wrapping key could be up to two years. In the case of keys used for only a single message or communications session, the cryptoperiod would be limited to a single communication session. Except for the latter, a maximum recipient-usage period of three years beyond the end of the originator-usage period is recommended.

8. *Symmetric RBG keys*:

a. Type Considerations: Symmetric RBG keys are used in deterministic random bit generation functions. The **approved** RBGs in [SP800-90] control key changes (e.g., during reseeding). The cryptoperiod consists of only an originator-usage period.

b. Cryptoperiod: Assuming the use of **approved** RBGs, the maximum cryptoperiod of symmetric RBG keys is determined by the design of the RBG (see [SP800-90]).

9. *Symmetric master key*:

a. Type Considerations: A symmetric master key (also called a key-derivation key) may be used multiple times to derive other keys using a (one-way) key-derivation function or method (see Section 8.2.4). Therefore, the cryptoperiod consists of only an originator-usage period for this key type. A suitable cryptoperiod depends on the nature and use of the keys derived from the master key and on considerations provided earlier in Section 5.3. The cryptoperiod of a key derived from a master key could be relatively short, e.g., a single use, communication session, or transaction. Alternatively, the master key could be used over a longer period of time to derive (or re-derive) multiple keys for the same or different purposes. The cryptoperiod of the derived keys depends on their use (e.g., as symmetric data-encryption or integrity authentication keys).

b. Cryptoperiod: An appropriate cryptoperiod for the symmetric master key might be one year, depending on its usage environment and the sensitivity/criticality of the information protected by the derived keys and the number of keys derived from the master key.

10. *Private key-transport key*:

a. Type Considerations: A private key-transport key may be used multiple times to decrypt keys. 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 corresponding public key-transport key need to be decrypted.

b. Cryptoperiod: Given 1) the use of **approved** algorithms and key sizes, 2) the volume of information that may be protected by keys encrypted under the corresponding public key-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 two years is recommended for the private key-transport key. In certain applications (e.g., email), where received messages are stored and decrypted at a later time, the cryptoperiod of the private key-transport key may exceed the cryptoperiod of the public key-transport key.

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. When the public key has been certified by a CA, the cryptoperiod ends when the *notAfter* date is reached on the last certificate issued for the public key.

Public key-transport keys can be publicly known. 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 corresponding private key.

b. Cryptoperiod: Based on cryptoperiod assumptions for the corresponding private keys, a recommendation for the maximum cryptoperiod might be about one to two years.

12. *Symmetric key-agreement key*:

a. Type Considerations: A symmetric key-agreement key may be used multiple times. The cryptoperiod of these keys depends on 1) environmental security factors, 2) the nature (e.g., types and formats) and volume of keys that are established, 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 **approved** algorithm and key scheme, 2) the cryptographic device meets [FIPS140] requirements, and 3) the risk levels are established in conformance to [FIPS199], an appropriate cryptoperiod for the key would be one to two years. In certain applications (e.g., email), where received messages are stored and decrypted at a later time, the recipient-usage period of the key may exceed the originator-usage period.

13. *Private static key-agreement key*:

a. Type Considerations: A private static (i.e., long-term) key-agreement key may be used multiple times. When the corresponding public key has been certified by a CA, the cryptoperiod ends when the *notAfter* date is reached on the last certificate issued for the public key.

As in the case of symmetric key-agreement keys, the cryptoperiod of these keys depends on 1) environmental security factors, 2) the nature (e.g., types and formats) and volume of keys that are established, 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 secret keying material.

b. Cryptoperiod: Given an assumption that the cryptography that employs private static key-agreement keys 1) employs an **approved** algorithm and key scheme, 2) the cryptographic device meets [FIPS140] requirements, and 3) the risk levels are established in conformance to [FIPS199], an appropriate cryptoperiod for the key would be one to two years. In certain applications (e.g., email), where received messages are stored and decrypted at a later time, the cryptoperiod of the private static key-agreement key may exceed the cryptoperiod of the corresponding public static key-agreement key.

14. *Public static key-agreement key*:

a. Type Considerations: The cryptoperiod for a public static (i.e., long-term) key-agreement key is usually the same as the cryptoperiod of the corresponding private static key-agreement key.

b. Cryptoperiod: The cryptoperiod of the public static key-agreement key may be one to two years.

15. *Private ephemeral key-agreement key*:

a. Type Considerations: Private ephemeral (i.e., short-term) key-agreement keys are the private key elements of asymmetric key pairs that are used in a single transaction 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 secret keying material.

b. Cryptoperiod: Private ephemeral key-agreement keys are used for a single key-agreement transaction. However, a private ephemeral key may be used multiple times to establish the same symmetric key with multiple parties during the same transaction (broadcast). The cryptoperiod of a private ephemeral key-agreement key is the duration of a single key-agreement transaction.

16. *Public ephemeral key-agreement key*:

a. Type Considerations: Public ephemeral (i.e., short-term) 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 used for a single key-agreement transaction. The cryptoperiod of the public ephemeral key-agreement key ends immediately after it is used to generate the shared secret. Note that in some cases, the cryptoperiod of the public ephemeral key-agreement key may be different for the participants in the key-agreement transaction. For example, consider an encrypted email application in which the email sender generates an ephemeral key-agreement key pair, and then uses the key pair to generate an encryption key that is used to encrypt the contents of the email. For the sender, the cryptoperiod of the public key ends when the shared secret is generated and the *encryption* key is derived. However, for the encrypted email receiver, the cryptoperiod of the ephemeral public key does not end until the shared secret is generated and the *decryption* key is determined; if the email is not processed immediately upon receipt (e.g., it is decrypted a week later than the email was sent), then the cryptoperiod of the ephemeral public key does not end (from the perspective of the receiver) until the shared secret is generated that uses that public key.

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. For this key type, the originator-usage period and the recipient-usage period are the same. 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 **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 no more 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 corresponding public key **shall** be the same.

b. Cryptoperiod: Given the use of **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 no more 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 corresponding private key.

b. Cryptoperiod: The cryptoperiod of the public authorization key **shall** be the same as the private authorization key: no more than two years.

Table 1 below is a summary of the cryptoperiods that are suggested for each key type. Longer or shorter cryptoperiods may be warranted, depending on the application and environment in which the keys will be used. However, when assigning a longer cryptoperiod than that suggested below, serious consideration **should** be given to the risks associated with doing so (see Section 5.3.1).

Table 1: Suggested cryptoperiods for key types[17]

Key Type	Cryptoperiod	
	Originator-Usage Period (OUP)	**Recipient-Usage Period**
1. Private Signature Key	1-3 years	–
2. Public Signature-Verification Key	Several years (depends on key size)	
3. Symmetric Authentication Key	≤ 2 years	≤ OUP + 3 years
4. Private Authentication Key	1-2 years	
5. Public Authentication Key	1-2 years	
6. Symmetric Data Encryption Keys	≤ 2 years	≤ OUP + 3 years
7. Symmetric Key Wrapping Key	≤ 2 years	≤ OUP + 3 years
8. Symmetric RBG Keys	See [SP800-90]	–
9. Symmetric Master Key	About 1 year	–
10. Private Key Transport Key	≤ 2 years[18]	
11. Public Key Transport Key	1-2 years	
12. Symmetric Key Agreement Key	1-2 years[19]	
13. Private Static Key Agreement Key	1-2 years[20]	

[17] In some cases, risk factors affect the cryptoperiod selection (see Section 5.3.1).

[18] In certain email applications where received messages are stored and decrypted at a later time, the cryptoperiod of the private key-transport key may exceed the cryptoperiod of the public key-transport key.

[19] In certain email applications where received messages are stored and decrypted at a later time, the key's recipient-usage period key may exceed the originator-usage period.

Key Type	Cryptoperiod	
	Originator-Usage Period (OUP)	Recipient-Usage Period
14. Public Static Key Agreement Key	1-2 years	
15. Private Ephemeral Key Agreement Key	One key-agreement transaction	
16. Public Ephemeral Key Agreement Key	One key-agreement transaction	
17. Symmetric Authorization Key	≤ 2 years	
18. Private Authorization Key	≤ 2 years	
19. Public Authorization Key	≤ 2 years	

5.3.7 Recommendations for Other Keying Material

Other keying material does not have well-established cryptoperiods, per se. The following recommendations 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 in its cryptographically protected form is no longer needed.
3. Shared secrets generated during the execution of key-agreement schemes **shall** be destroyed as soon as they are no longer needed to derive keying material.
4. RBG seeds **shall** be destroyed immediately after use.
5. Other public information **should not** be retained longer than needed for cryptographic processing.
6. Other secret information **shall not** be retained longer than necessary.
7. Intermediate results **shall** be destroyed immediately after use.

5.4 Assurances

When cryptographic keys and domain parameters are stored or distributed, they may pass through unprotected environments. In this case, specific assurances are required before the key or domain parameters may be used to perform normal cryptographic operations.

[20] In certain email applications whereby received messages are stored and decrypted at a later time, the cryptoperiod of the private static key-agreement key may exceed the cryptoperiod of the public static key-agreement key.

5.4.1 Assurance of Integrity (Integrity Protection)

Assurance of integrity **shall** be obtained prior to using all keying material.

At a minimum, assurance of integrity **shall** be obtained by verifying that the keying material has the appropriate format and came from an authorized source. Additional assurance of integrity may be obtained by the proper use of error detection codes, message authentication codes, and digital signatures.

5.4.2 Assurance of Domain Parameter Validity

Domain parameters are used by discrete log public-key algorithms during the generation of key pairs and digital signatures, and during the generation of shared secrets (during the execution of a key-agreement scheme) that are subsequently used to derive keying material. Assurance of the validity of the domain parameters is important to applications of public-key cryptography and **shall** be obtained prior to using them.

Invalid domain parameters could void all intended security for all entities using the domain parameters. Methods for obtaining assurance of domain-parameter validity for the DSA and ECDSA digital signature algorithms are provided in [SP800-89]. Methods for obtaining assurance of domain-parameter validity for finite-field and elliptic-curve discrete-log key-agreement algorithms are provided in [SP800-56A].

Note that if a public key is certified by a CA for these algorithms, the CA could obtain this assurance during the certification process. Otherwise, the key-pair owner and any relying parties are responsible for obtaining the assurance.

5.4.3 Assurance of Public-Key Validity

Assurance of public-key validity **shall** be obtained on all public keys before using them.

Assurance of public-key validity gives the user confidence that the public key is arithmetically correct. This reduces the probability of using weak or corrupted keys. Invalid public keys could result in voiding the intended security, including the security of the operation (i.e., digital signature, key establishment, or 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). One of several ways to obtain assurance of validity is for an entity to verify certain mathematical properties that the public key should have. Another way is to obtain the assurance from a trusted third party (e.g., a CA) that the trusted party validated the properties.

Methods of obtaining assurance of public-key validity for the DSA, ECDSA and RSA digital signature algorithms are provided in [SP800-89]. Methods for obtaining this assurance for the finite-field and elliptic-curve discrete-log key-establishment schemes are provided in [SP800-56A]. Methods for obtaining assurance of (partial) public-key validity for the RSA key-establishment schemes are provided in [SP800-56B].

5.4.4 Assurance of Private-Key Possession

Assurance of static (i.e., long-term) private-key possession **shall** be obtained before the use of the corresponding static public key. Assurance of validity **shall** always be obtained prior to, or concurrently with, assurance of possession. Assurance of private-key possession **shall** be obtained by both the owner of the key pair and by other entities that receive the public key of that key pair and use it to interact with the owner.

For specific details regarding assurance of the possession of private key-establishment keys, see [SP800-56A] and [SP800-56B]; for specific details regarding assurance of the possession of private digital-signature keys, see [SP800-89]. Note that for public keys that are certified by a CA, the CA could obtain this assurance during the certification process. Otherwise, the owner and relying parties are responsible for obtaining the assurance.

5.5 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 Section 6), 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 cryptographic protection to information (e.g., compute a digital signature or encrypt information) **shall** cease, and the compromised key **shall** be revoked (see Section 8.3.5). However, the continued use of the key under controlled circumstances 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 [SP800-57, Part 2]). The continued use of a compromised key **shall** be limited to processing already-protected information. In this case, the entity that uses the information **shall** be made fully aware of the dangers involved. 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. The unauthorized disclosure 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 unauthorized disclosure of a key used to provide confidentiality protection[21] (i.e., via encryption) means that all information encrypted by that key could be determined by unauthorized entities. 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. In addition, a compromised key could be used by an adversary to encrypt information of the adversary's choosing, thus providing false information.

   The unauthorized disclosure of a private signature key means that the integrity and non-repudiation qualities of all data signed by that key are suspect. An unauthorized party in possession of the private key could sign false information and make it appear to be valid. In cases where it can be shown that the signed data was protected by other mechanisms (e.g., physical security) from a time before the compromise, the signature may still have some value. For example, if a signed message was received on day 1,

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

and it was later determined that the private signing key was compromised on day 15, the receiver may still have confidence that the message is valid because it was maintained in the receiver's possession before day 15. Note that cryptographic timestamping may also provide protection for messages signed before the private signature key was compromised. However, the security provided by these other mechanisms is now critical to the security of the signature. In addition, the non-repudiation of the signed message may be questioned, since the private signature key may have been disclosed to the message receiver, who then altered the message in some way.

The disclosure of a CA's private signature key means that an adversary can create fraudulent certificates and Certificate Revocation Lists (CRLs).

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 substitution or modification of a key used to provide integrity[22] calls into question the integrity of all information protected by the key.

3. A compromise of a key's usage or application association means that the key could be used for the wrong purpose (e.g., for 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 does not know who the other entity really is).

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.

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

c. Restricting plaintext symmetric and private keys to physically protected containers. This includes key generators, key-transport devices, key loaders, cryptographic modules, and key-storage devices.

d. Using integrity checks to ensure 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 wrapped key or to the key's metadata will be detected.

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

e. Employing key confirmation (see Section 4.2.5.5) 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., using a MAC or a digital signature).

h. The use of trusted timestamps for signed data.

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

j. Creating a compromise-recovery plan, especially in the case of the compromise of a CA key.

The worst form of key compromise is one that is not detected. Nevertheless, even in this case, certain protective measures can be taken. Cryptographic Key Management Systems (CKMSs) **should** be designed to mitigate the negative effects of a key compromise. A CKMS **should** be designed so that the compromise of a single key compromises as little data 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 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 [SP800-57, 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 key compromise 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 certificate. 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 method for obtaining a new key (i.e., re-keying),
4. An inventory of all cryptographic keys (e.g., the location of all certificates in a system),
5. The education of all appropriate personnel on the recovery procedures,
6. An identification of all personnel needed to support the recovery procedures,
7. Policies that key-revocation checking be enforced (to minimize the effect of a compromise),

8. The monitoring of the re-keying operations (to ensure that all required operations are performed for all affected keys), and
9. Any other recovery procedures.

Other compromise-recovery procedures may include:

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

5.6 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.

5.6.1 Comparable Algorithm Strengths

Cryptographic algorithms can provide different "strengths" of security, depending on the algorithm and the key size used (when a key is employed). Table 2 gives the current estimates for the maximum security strengths that the **approved** symmetric and asymmetric cryptographic algorithms can provide, given keys of a specified length. These estimates were made under the assumption that the keys used with those algorithms are generated and handled in accordance with specific rules (e.g., the keys are generated using RBGs that were seeded with sufficient entropy). However, these rules are often not followed, and the security provided to the data protected by those keys may be somewhat less than the security strength estimates provided

Two algorithms are considered to be of comparable 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 and sufficient entropy) is approximately the same using a given resource. The security strength of an algorithm 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 estimated maximum security strength is comparable to a symmetric algorithm with an *X*-bit key is said have an "estimated maximum security strength of *X* bits" or to be able to provide "*X* bits of security". Given a few plaintext blocks and corresponding ciphertext, an algorithm that can provide *X* bits of security would, on average, take $2^{X-1}T$ units of time to attack, where *T* is the amount of time that is required to perform one encryption of a plaintext value and compare the result against the corresponding ciphertext value.

Determining the security strength of an algorithm can be nontrivial. For example, consider TDEA, which uses three 56-bit keys (K_1, K_2 and K_3). If each of these keys is independently generated, then this is called three-key TDEA (3TDEA). However, if K_1 and K_2 are independently generated, and K_3 is set equal to K_1, then this is called two-key TDEA (2TDEA). One might expect that 3TDEA would provide $56 \times 3 = 168$ bits of strength. However, there is an attack on 3TDEA that reduces the strength to the work that would be involved in exhausting a 112-bit key. For 2TDEA, if exhaustion were the best attack, then the strength of 2TDEA 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, the security strength of 2TDEA decreases as the number of matched plaintext/ciphertext pairs increases. If the attacker can obtain approximately 2^{40} such pairs and has sufficient memory and computational power, then 2TDEA can provide an estimated maximum security strength of about 80 bits; if the attacker has 2^{56} plaintext/ciphertext pairs, with significantly more memory and computational power, then the estimated maximum security strength would be about 56 bits.

The comparable key-size classes discussed in this section are based on estimates made as of the publication of this Recommendation using currently known methods. 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. If quantum attacks become practical, 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.

The use of strong cryptographic algorithms may mitigate security issues other than just brute-force cryptographic attacks. The algorithms may unintentionally be implemented in a manner that leaks small amounts of information about the key. In this case, the larger key may reduce the likelihood that this leaked information will eventually compromise the key.

When selecting a block-cipher cryptographic algorithm (e.g., AES or TDEA), 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 [SP800-38] is dependent on the block size. More information on this issue is provided in [SP800-38].

Table 2 provides estimated, comparable maximum security strengths for the **approved** algorithms and key lengths.

1. Column 1 indicates the estimated maximum security strength (in bits) provided by the algorithms and key sizes in a particular row. Note that the security strength is not necessarily the same as the length of the key for the algorithms in the other columns, due to attacks on those algorithms that provide computational advantages.

2. Column 2 identifies the symmetric-key algorithms that can provide the security strength indicated in column 1, where 2TDEA and 3TDEA are specified in [SP800-67], and AES is specified in [FIPS197]. 2TDEA is TDEA with two different keys; 3TDEA is TDEA with three different keys.

3. Column 3 indicates the minimum size of the parameters associated with the standards that use finite-field cryptography (FFC). Examples of such algorithms include DSA, as defined in [FIPS186] for digital signatures, and Diffie-Hellman (DH) and MQV key agreement, as defined in [SP800-56A], where L is the size of the public key, and N is the size of the private key.

4. Column 4 indicates the value for *k* (the size of the modulus *n*) for algorithms based on integer-factorization cryptography (IFC). The predominant algorithm of this type is the RSA algorithm. RSA is **approved** in [FIPS186] for digital signatures, and in [SP800-56B] for key establishment. The value of *k* is commonly considered to be the key size.
5. Column 5 indicates the range of *f* (the size of *n*, where *n* is the order of the base point *G*) for algorithms based on elliptic-curve cryptography (ECC) that are specified for digital signatures in [ANSX9.62] and adopted in [FIPS186], and for key establishment as specified in [SP800-56A]. The value of *f* is commonly considered to be the key size.

Table 2: Comparable strengths

Security Strength	Symmetric key algorithms	FFC (e.g., DSA, D-H)	IFC (e.g., RSA)	ECC (e.g., ECDSA)
≤ 80	2TDEA[23]	L = 1024 N = 160	k = 1024	f = 160-223
112	3TDEA	L = 2048 N = 224	k = 2048	f = 224-255
128	AES-128	L = 3072 N = 256	k = 3072	f = 256-383
192	AES-192	L = 7680 N = 384	k = 7680	f = 384-511
256	AES-256	L = 15360 N = 512	k = 15360	f = 512+

Note that the 192-bit and 256-bit key strengths identified for the FFC and IFC algorithms (shaded in yellow) are not currently included in the NIST standards for interoperability and efficiency reasons.

Also, note that algorithm/key-size combinations that have been estimated at a maximum security strength of less than 112 bits (shaded in orange above) are no longer approved for applying cryptographic protection on Federal government information (e.g., encrypting data or generating a digital signature). However, some flexibility is allowed for processing already-protected information at those security strengths (e.g., decrypting encrypted data or verifying digital signatures), if the receiving entity accepts the risks associated with doing so. See [SP800131A] for more detailed information.

Appropriate hash functions that may be employed will be determined by the algorithm, scheme or application in which the hash function is used and by the minimum security-strength to be provided. Table 3 lists the **approved** hash functions specified in [FIPS186] and [FIPS202] that

[23] See the example in the third paragraph of Section 5.6.1.

can be used to provide each identified security strength for various hash-function applications: digital signatures, HMAC, key derivation and random bit generation.

Table 3: Hash function that can be used to provide the targeted security strengths

Security Strength	Digital Signatures and hash-only applications	HMAC[24], Key Derivation Functions[25], Random Number Generation[26]
≤ 80	SHA-1[27]	
112	SHA-224, SHA-512/224, SHA3-224	
128	SHA-256, SHA-512/256, SHA3-256	SHA-1
192	SHA-384, SHA3-384	SHA-224, SHA-512/224
≥ 256	SHA-512, SHA3-512	SHA-256, SHA-512/256, SHA-384, SHA-512, SHA3-512

Note that some security strengths in the table do not indicate a hash function for the application; it is always acceptable to use a hash function with a higher estimated maximum security strength than that required for the application.

Note that in the case of HMAC, which requires a key, the estimate assumes that a key whose length and entropy are at least equal to the security strength is used.

For some applications, a cryptographic key is associated with the application and needs to be considered when determining the security strength actually afforded by the application. For example, for the generation of digital signatures, the minimum key length for the keys for a given security strength is provided in the FFC, IFC and ECC columns of Table 2; while for HMAC, the key lengths are discussed in [SP800-107].

Note that hash functions and applications providing less than 112 bits of security strength (shaded in orange) are no longer approved for applying cryptographic protection on Federal government information (e.g., generating a digital signature). However, some flexibility is allowed for processing already-protected information at those security strengths (e.g., verifying digital signatures), if the receiving entity accepts the risks associated with doing so. See [SP800131A] for more detailed information.

[24] Assumes that pre-image resistance is required, rather than collision resistance.

[25] The security strength for key-derivation assumes that the shared secret contains sufficient entropy to support the desired security strength.

[26] The security strength assumes that the random number generator has been provided with adequate entropy to support the desired security strength.

[27] SHA-1 has been demonstrated to provide less than 80 bits of security for digital signatures, which require collision resistance; at the publication of this Recommendation, the security strength against digital signature collisions remains the subject of speculation.

5.6.2 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 TDEA).

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 4 provides general recommendations that may be used to select an appropriate suite of algorithms and key sizes for Federal Government unclassified applications to protect sensitive data. A schedule for increasing the security strengths for applying cryptographic protection to data (e.g., encrypting or digitally signing) is specified in the table. Transition details for algorithms, key sizes and applications are provided in [SP800-131A]. The table is organized as follows:

1. Column 1 is divided into two sub-columns. The first sub-column indicates the security strength to be provided; the second sub-column indicates whether cryptographic protection is being applied to data (e.g., encrypted), or whether cryptographically protected data is being processed (e.g., decrypted).

2. Columns 2 and 3 indicate the time frames during which the security strength is either acceptable, OK for legacy use or disallowed[28].

   - "Acceptable" indicates that the algorithm or key length is not known to be insecure.
   - "Legacy-use" means that an algorithm or key length may be used because of its use in legacy applications (i.e., the algorithm or key length can be used to process cryptographically protected data).
   - "Disallowed" means that an algorithm or key length **shall not** be used for applying cryptographic protection.

See [SP800-131A] for specific details and for any exceptions to the general guidance provided in Table 4.

Table 4: Security-strength time frames

<table>
<tr><th colspan="2">Security Strength</th><th>Through 2030</th><th>2031 and Beyond</th></tr>
<tr><td rowspan="2">≤ 80</td><td>Applying</td><td colspan="2">Disallowed</td></tr>
<tr><td>Processing</td><td colspan="2">Legacy-use</td></tr>
<tr><td>112</td><td>Applying</td><td>Acceptable</td><td>Disallowed</td></tr>
</table>

[28] A fourth category − deprecated − was used in the previous version of this Recommendation, but is not currently being used.

Security Strength		Through 2030	2031 and Beyond
	Processing		Legacy use
128	Applying/Processing	Acceptable	Acceptable
192		Acceptable	Acceptable
256		Acceptable	Acceptable

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 period **shall** be used for applying cryptographic protection (e.g., encryption). The following examples are provided to clarify the use of the table:

1. If information is cryptographically protected (e.g., digitally signed) in 2015, and the maximum-expected security life of that data is only one year, any of the **approved** digital-signature algorithms or key sizes that provide at least 112 bits of security strength may be used.
2. If the information is to be digitally signed in 2025, and the expected security life of the data is six years, then an algorithm or key size that provides at least 128 bits of security strength is required.

5.6.3 Using Algorithm Suites

Algorithm suites that combine algorithms with a mixture of estimated maximum security strengths is 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 to the data to be protected. In general, the weakest algorithm and key size used to provide cryptographic protection determines the strength of the protection. A determination of the actual strength of the protection provided for 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 the details of how the key was generated (e.g., the security strength supported by the RBG used during the generation of the key) and how the key was handled subsequent to generation (e.g., was the key wrapped by an algorithm with a security strength less than the security strength intended for the key's use.

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

1. When a key-establishment scheme is used to establish keying material for use with one or more algorithms (e.g., TDEA, AES, or HMAC), the security strength that can be supported by the keying material is determined by the weakest algorithm and key size used. For example, if a 224-bit ECC key is used as specified in [SP80056A] to establish a 128-bit AES key, no more than 112 bits of security can be provided for any information protected by that AES key, since the 224-bit ECC can only provide a maximum of 112 bits of security.
2. When a hash function and digital signature algorithm are used in combination to compute a digital signature, the security strength of the signature is determined by the

weaker of the two processes. For example, if SHA-256 is used with RSA and a 2048-bit key, the combination can provide no more than 112 bits of security, because a 2048-bit RSA key cannot provide more than 112 bits of security strength.

3. When a random bit generator is used to generate a key for a cryptographic algorithm that is intended to provide *X* bits of security, an **approved** random bit generator **shall** be used that provides at least *X* bits of security.

If it is determined that a specific level of security is required for the protection of data, then an algorithm and key size suite needs to be selected that could provide that level of security (as a minimum). For example, if 128 bits of security are required for data that is to be communicated and provided with confidentiality protection, and integrity and source authentication, 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 performance may be a little slower.

b. Integrity authentication and source authentication: If only one cryptographic operation is preferred, use digital signatures. SHA-256 or a larger hash function could be used. Select an algorithm for digital signatures from what is available to an application (e.g., ECDSA 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 that is based on the application and environment (see [SP800-56A] or [SP800-56B]), the availability of an algorithm in an implementation, and its performance. Select a key size from Table 2 for an algorithm and key size that can provide at least 128 bits of security. For example, if an ECC key-agreement scheme is available, use an ECC scheme with at least a 256-bit key (the value of *f* in Table 2). However, the key used for key agreement **shall** be different from the ECDSA 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 **should** ensure that the algorithms and key sizes can be readily updated.

5.6.4 Transitioning to New Algorithms and Key Sizes

The estimated time period during which data protected by a specific cryptographic algorithm (and key size) remains secure is called the *algorithm security lifetime*. During this time, the algorithm may be used to both apply cryptographic protection (e.g., encrypt data) and to process the protected information (e.g., decrypt data); the algorithm is expected to provide adequate protection for the protected data during this period.

Typically, an organization selects the cryptographic services that are needed for a particular application. Then, based on the algorithm security lifetime and the security life of the data to be protected, an algorithm and key-size suite is selected that is sufficient to meet the requirements. The organization then establishes a key-management system (if required), including validated cryptographic products that provide the services required by the application. As an algorithm and/or key-size suite nears the end of its security lifetime, transitioning to a new algorithm and key-size suite **should** be planned.

When the algorithm or key size is determined to no longer provide the desired protection for information (e.g., the algorithm may have been "broken"), any information protected by the algorithm or key size is considered to be suspect (e.g., the data may no longer be confidential, or the integrity cannot be assured). If the protected data is retained, it **should** be re-protected using an **approved** algorithm and key size that will protect the information for the remainder of its security life. However, it **should** be assumed that encrypted information could have been collected and retained by unauthorized entities (adversaries) for decryption at some later time. In addition, the recovered plaintext could be used to attempt a matched plaintext-ciphertext attack on the new algorithm.

When using Tables 2, 3 and 4 to select the appropriate algorithm and key size, it is very important to take the expected security life of the data into consideration. As stated earlier, an algorithm (and key size) may be used both to apply cryptographic protection to data and process the protected data. When the security life of the data is taken into account, cryptographic protection **should not** be applied to data using a given algorithm (and key size) if the security life of the data extends beyond the end of the algorithm security lifetime (i.e., into the timeframe when the algorithm or key size is disallowed; see Table 4). The period of time that an algorithm (and key size) may be used to apply cryptographic protection is called the *algorithm originator-usage period*. The algorithm security life = (the algorithm usage period + the security life of the data) (see Figure 2).

For example, suppose that 3TDEA is to be used to provide confidentiality protection for data with a security life of four years. Table 2 indicates that 3TDEA has a maximum security strength of 112 bits. Table 4 indicates that an algorithm with a security strength of 112 bits has an algorithm security lifetime that extends through 2030 for applying cryptographic protection (i.e., encryption, in this case), but not beyond. Since the data has a four-year security life, the algorithm originator-usage period must end by December 31 2026 (rather than 2030) in order to ensure that all data protected by 3TDEA is secure during its entire security life (i.e., the algorithm could not be used to encrypt data beyond 2026). See Figure 2. After 2026, the algorithm could be used to decrypt data for another four years, with the expectation that the confidentiality of the data continues to be protected at a security strength of 112 bits. If the security life of the data was estimated correctly, the data would no longer need this confidentiality protection after 2030. However, if the security life of the data is longer than originally expected, then the protection provided after 2030 may be less than required, and there is some risk that the confidentiality of the data may be compromised (after 2030); accepting the risk associated with the possible compromise is indicated by the "legacy use" indication in Table 4.

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 minimize costly transitions. However, it should be noted that selecting some algorithms or key sizes that are unnecessarily large might have adverse performance effects (e.g., the algorithm may be unacceptably slow).

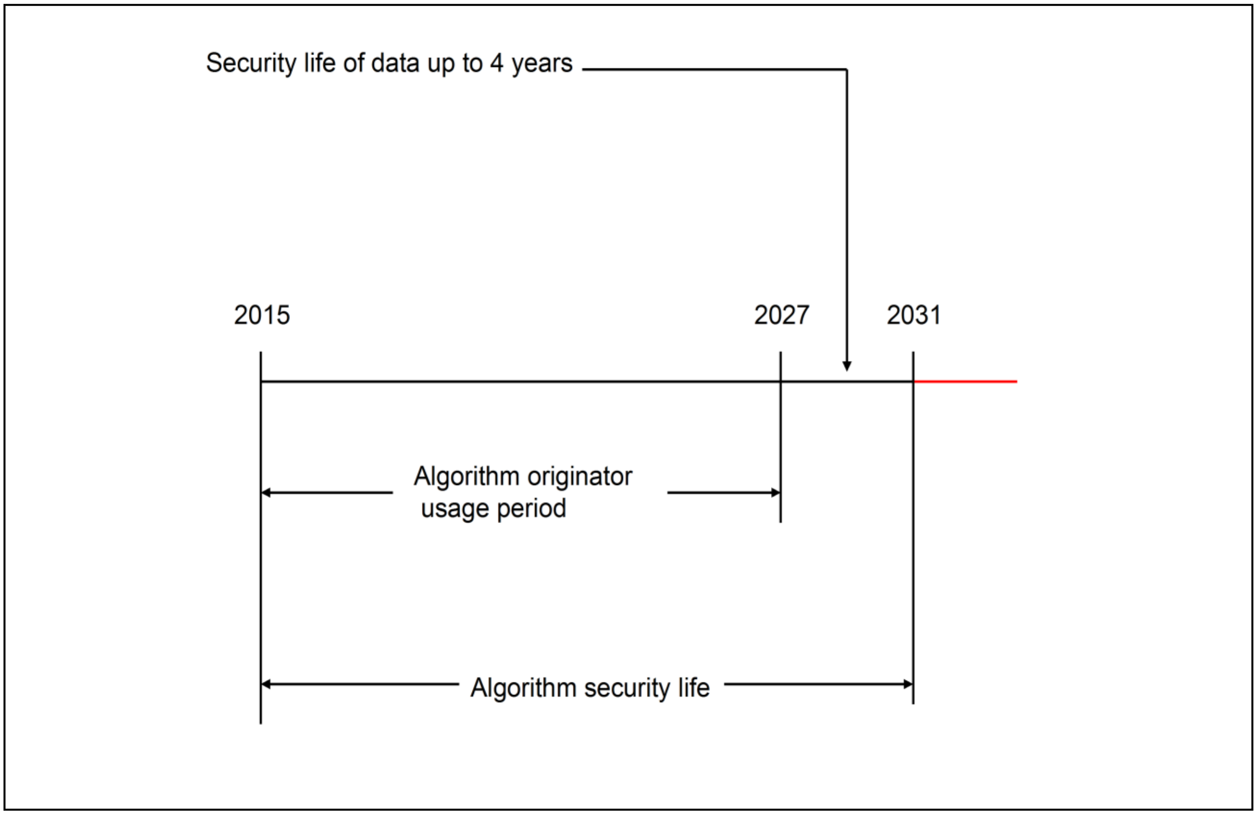

Figure 2: Algorithm Originator-Usage Period Example

The process of transitioning to a new algorithm or a new key size may be as simple as selecting a more secure option in the security suites offered by the current system, or it can be as complex as building a whole new system. However, given that it is necessary to develop a new algorithm suite for a system, the following issues should be considered.

1. **The sensitivity of information and the system lifetime:** The sensitivity of the information that will need to be protected by the system for the lifetime of the new algorithm(s) should be evaluated in order to determine the minimum security-requirement for the system. Care should be taken not to underestimate the required lifetime of the system or the sensitivity of information that it may need to protect. Many decisions that were initially considered as temporary or interim decisions about data sensitivity have since been proven to be inadequate (e.g., the sensitivity of the information lasted well beyond its initially expected lifetime).

2. **Algorithm selection:** New algorithms should be carefully selected to ensure that they meet or exceed the minimum security-requirement of the system. In general, it is relatively easy to select cryptographic algorithms and key sizes that offer high security. However, it is wise for the amateur to consult a cryptographic expert when making such decisions. Systems **should** offer algorithm-suite options that provide for future growth.

3. **System design:** A new system **should** be designed to meet the minimum performance and security requirements. This is often a difficult task, since

performance and security goals may conflict. All aspects of security (e.g., physical security, computer security, operational security, and personnel security) are involved. If a current system is to be modified to incorporate new algorithms, the consequences need to be analyzed. For example, the existing system may require significant modifications to accommodate the footprints (e.g., key sizes, block sizes, etc.) of the new algorithms. In addition, the security measures (other than the cryptographic algorithms) retained from the current system **should** be reviewed to assure that they will continue to be effective in the new system.

4. **Pre-implementation evaluation:** Strong cryptography may be poorly implemented. Therefore, a changeover to new cryptographic techniques **should not** be made without an evaluation as to how effective and secure they are in the system.
5. **Testing:** Any system **should** be tested before it is employed.
6. **Training:** If the new system requires that new or different tasks (e.g., key management procedures) be performed, then the individuals who will perform those tasks **should** be properly trained. Features that are thought to be improvements may be viewed as annoyances by an untrained user.
7. **System implementation and transition:** Care **should** be taken to implement the system as closely as possible to the design. Exceptions **should** be noted.
8. **Transition:** A transition plan **should** be developed and followed so that the changeover from the old to the new system runs as smoothly as possible.
9. **Post-implementation evaluation**: The system **should** be evaluated to verify that the implemented system meets the minimum security requirements.

5.6.5 Security Strength Reduction

At some time, the security strength provided by an algorithm or key may be reduced or lost completely. For example, the algorithm or key length used may no longer offer adequate security because of improvements in computational capability or cryptanalysis. In this case, applying protection to “new” information can be performed using stronger algorithms or keys. However, information that was previously protected using these now-inadequate algorithms and keys may no longer be secure. This information may include other keys, or other sensitive data protected by the keys. A reduction in the security strength provided by an algorithm or key has the following implications:

- Encrypted information: The security of encrypted information that was available at any time to unauthorized entities in its encrypted form should be considered suspect. For example, keys that were transmitted in encrypted form (e.g., using a key-wrapping key or key-transport key and an algorithm or key length that is later broken) may need to be considered as compromised, since an adversary could have saved the encrypted form of the keys for later decryption in case methods for breaking the algorithm would eventually be found (see Section 5.5 for a discussion of key compromise). Even if the transmitted, encrypted information is subsequently re-encrypted for storage using a different key or algorithm, the information may already be compromised because of the weakness of the transmission algorithm or key.

Encrypted information that was not "exposed" in this manner (e.g., not transmitted) may still be secure, even though the encryption algorithm or key length no longer provides adequate protection. For example, if the encrypted form of the keys and the information protected by those keys was never transmitted, then the information may still be confidential.

The lessons to be learned are that an encryption mechanism used for information that will be available to unauthorized entities in its encrypted form (e.g., via transmission) should provide a high level of security protection, and the use of each key should be limited (i.e., the cryptoperiod should be short) so that a compromised key cannot be used to reveal very much information. If the algorithm itself is broken[29], an adversary is forced to perform more work when each key is used to encrypt a very limited amount of information in order to decrypt all of the information. See Section 5.3.6 for a discussion about cryptoperiods.

- Digital signatures on stored data[30]: Digital signatures may be computed on data prior to transmission and subsequent storage. In this case, both the signed data and the digital signature would be stored. If the security strength of the signature is later reduced (e.g., because of a break of the algorithm), the signature may still be valid if the stored data and its associated digital signature have been adequately protected from modification since a time prior to the reduction in strength (e.g., by applying a digital signature using a stronger algorithm or key). See Section 5.5, item 1 for further discussion. Storage capabilities are being developed that employ cryptographic timestamps to store digitally signed data beyond the normal security life of the original signature mechanism or its keys.
- Symmetric authentication codes on stored data[31]: Like digital signatures, symmetric authentication codes (i.e., MACs) may be computed on data prior to transmission and subsequent storage. If the received data and authentication code are stored as received, and the security strength of the authentication algorithm or key is later reduced (e.g., because of a break of the algorithm), the authentication code may still be valid if the stored data and its associated authentication code have been adequately protected from modification since a time prior to the reduction in strength (e.g., by applying another authentication code using a stronger algorithm or key). See Section 5.5, item 1 for further discussion. Storage capabilities are being developed that employ cryptographic timestamps to store authenticated data beyond the normal security life of the original authentication mechanism or its keys.

[29] It is easier to recover a key than exhaustive search.

[30] Digital signatures on data that is transmitted, but not stored are not considered, as their value is considered to be short-lived, e.g., the digital signature was intended to be used to detect errors introduced only during transmission.

[31] Symmetric authentication codes on data that is transmitted, but not stored are not considered, as their value is considered to be short-lived.

6 Protection Requirements for Cryptographic Information

This section gives guidance on the types of protection required for keying material. Cryptographic keying material is defined as the cryptographic key and associated information required to use the key (i.e., the metadata). 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." A FIPS 140-validated cryptographic module may provide much of the protection needed; however, whenever the keying material exists external to a [FIPS140] cryptographic module, 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. [SP800-152] provides guidance for Federal Cryptographic Key Management Systems (FCKMSs) on the protection of keys and metadata when outside a FIPS 140-validated cryptographic module, as well as other key management factors to be addressed.

6.1 Protection and Assurance Requirements

Keying material **should** be (operationally) available as long as the associated cryptographic service is required. Keys may be maintained within a cryptographic module while they are being actively used, or they may be stored externally (provided that proper protection is afforded) and recalled as needed. Some keys may need to be archived if required beyond the key's originator-usage period (see Section 5.3.5).

The following protections and assurances may be required for the keying material.

Integrity protection **shall** be provided for all keying material. Integrity protection always involves checking the source and format of received keying material (see Section 5.4.1). When the key exists within a validated cryptographic module, appropriate integrity protection is provided when the cryptographic module conforms to [FIPS140], at a security level that is consistent with the [FIPS 199] impact level associated with the data to be protected by the key (see [SP800-152]). When a key is available outside a cryptographic module, integrity protection **shall** be provided by appropriate cryptographic integrity mechanisms (e.g. cryptographic checksums, cryptographic hash functions, MACs, and digital signatures), non-cryptographic integrity mechanisms (e.g. CRCs, parity checks, etc.) (see Appendix A), or physical protection mechanisms. Guidance for the selection of appropriate integrity mechanisms is given in Sections 6.2.1.2 and 6.2.2.2.

C*onfidentiality protection* for all symmetric and private keys **shall** be provided. Public keys generally do not require confidentiality protection. When the symmetric or private key exists within a validated cryptographic module, appropriate confidentiality protection is provided when the cryptographic module conforms to [FIPS140], at a security level that is consistent with the [FIPS199] impact level associated with the data to be protected by the key (see [SP800-152]). When a symmetric or private key is available outside a cryptographic module, confidentiality protection **shall** be provided either by encryption (e.g., key wrapping) at an appropriate security strength (see [SP800-152]), by the use of separate key components (see Section 6.2.1.3) 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 6.2.1.3 and 6.2.2.3.

Association protection **shall** be provided for 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 6.2.1.4 and 6.2.2.4.

Assurance of domain-parameter and public-key validity provides confidence that the parameters and keys are arithmetically correct (see Sections 5.4.2 and 5.4.3). Guidance for the selection of appropriate validation mechanisms is given in [SP800-56A] and [SP800-89], as well as in this document.

Assurance of private key possession provides assurance that the owner of a public key actually possesses the corresponding private key (see Section 5.4.4).

The *period of protection* for cryptographic keys, associated key information, and cryptographic parameters (e.g. initialization vectors) 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 includes the cryptoperiod of the key (see Section 5.3). The period of protection is not necessarily the same for integrity as it is for confidentiality. Integrity protection may only be required until a key is no longer used (but not yet destroyed), but confidentiality protection may be required until the key is actually destroyed.

6.1.1 Summary of Protection and Assurance Requirements for Cryptographic Keys

Table 5 provides a summary of the protection requirements for keys during distribution and storage. Methods for providing the necessary protection are discussed in Section 6.2.

Guide to Table 5:

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. In some cases, the word "support" is used in this column. This means that the associated key is used to support the primary cryptographic services of confidentiality, integrity authentication, and source authentication. For example, a key-agreement key may support a confidentiality service by establishing the key used to provide confidentiality; an RBG key is used to provide the random values for generating the keys to be used to generate digital signatures.

c. Column 3 (Security Protection) indicates the type of protection required for the key (i.e., integrity and confidentiality).

d. Column 4 (Association Protection) indicates the types of associations that need to be protected for that key, such as associating the key with the usage or application, the authorized communications participants or other indicated information. The association with domain parameters applies only to algorithms where they are used.

e. Column 5 (Assurances Required) indicates whether assurance of public-key validity and/or assurance of private-key possession needs to be obtained as defined in [SP800-56A], [SP800-56B], [SP800-89] and this Recommendation. Assurance of public-key validity provides a degree of confidence that a key is arithmetically correct. See Section 5.4.3 for further details. Assurance of private-key possession provides a degree of confidence that the entity providing a public key actually possessed the associated private key at some time. See Section 5.4.4 for further details.

f. Column 6 (Period of Protection) indicates the length of time that the integrity and/or confidentiality of the key needs to be maintained (see Section 5.3). Symmetric keys and private keys **shall be** destroyed at the end of their period of protection (see Sections 8.3.4 and 9.3).

Table 5: Protection requirements for cryptographic keys

Key Type	Security Service	Security Protection	Association Protection	Assurances Required	Period of Protection
Private signature key	Source authentication; Integrity authentication; Support non-repudiation	Integrity[32]; Confidentiality	Usage or application; Domain parameters; Public signature-verification key	Possession	From generation until the end of the cryptoperiod
Public signature-verification key	Source authentication; Integrity authentication; Support non-repudiation	Integrity;	Usage or application; Key pair owner Domain parameters; Private signature key; Signed data	Validity	From generation until no protected data needs to be verified
Symmetric authentication key	Source authentication; Integrity authentication	Integrity; Confidentiality	Usage or application; Other authorized entities; Authenticated data		From generation until no protected data needs to be verified
Private authentication key	Source authentication; Integrity authentication	Integrity; Confidentiality	Usage or application; Public authentication key; Domain parameters	Possession	From generation until the end of the cryptoperiod
Public authentication key	Source authentication; Integrity authentication	Integrity	Usage or application; Key pair owner; Authenticated data; Private authentication key; Domain parameters	Validity	From generation until no protected data needs to be authenticated

[32] Integrity protection can be provided by a variety of means. See Sections 6.2.1.2 and 6.2.2.2.

Symmetric data-encryption/ decryption key	Confidentiality	Integrity; Confidentiality	Usage or application; Other authorized entities; Plaintext/Encrypted data		From generation until the end of the lifetime of the data or the end of the cryptoperiod, whichever comes later
Symmetric key-wrapping key	Support	Integrity; Confidentiality	Usage or application; Other authorized entities; Encrypted keys		From generation until the end of the cryptoperiod or until no wrapped keys require protection, whichever is later.
Symmetric RBG keys	Support	Integrity; Confidentiality	Usage or application		From generation until replaced
Symmetric master key	Support	Integrity; Confidentiality	Usage or application; Other authorized entities; Derived keys		From generation until the end of the cryptoperiod or the end of the lifetime of the derived keys, whichever is later.
Private key-transport key	Support	Integrity; Confidentiality	Usage or application; Encrypted keys; Public key-transport key	Possession	From generation until the end of the period of protection for all transported keys
Public key-transport key	Support	Integrity	Usage or application; Key pair owner; Private key-transport key	Validity	From generation until the end of the cryptoperiod
Symmetric key-agreement key	Support	Integrity; Confidentiality	Usage or application; Other authorized entities		From generation until the end of the cryptoperiod or until no longer needed to determine a key, whichever is later
Private static key-agreement key	Support	Integrity; Confidentiality	Usage or application; Domain parameters; Public static key-agreement key	Possession	From generation until the end of the cryptoperiod or until no longer needed to determine a key, whichever is later

Public static key-agreement key	Support	Integrity	Usage or application; Key pair owner; Domain parameters; Private static key-agreement key	Validity	From generation until the end of the cryptoperiod or until no longer needed to determine a key, whichever is later
Private ephemeral key-agreement key	Support	Integrity; Confidentiality	Usage or application; Public ephemeral key-agreement key; Domain parameters;		From generation until the end of the key-agreement process After the end of the process, the key **shall** be destroyed
Public ephemeral key-agreement key	Support	Integrity[33]	Key pair owner; Private ephemeral key-agreement key; Usage or application; Domain parameters	Validity	From generation until the key-agreement process is complete
Symmetric authorization keys	Authorization	Integrity; Confidentiality	Usage or application; Other authorized entities		From generation until the end of the cryptoperiod of the key
Private authorization key	Authorization	Integrity; Confidentiality	Usage or application; Public authorization key; Domain parameters	Possession	From generation until the end of the cryptoperiod of the key
Public authorization key	Authorization	Integrity	Usage or application; Key pair owner; Private authorization key; Domain parameters	Validity	From generation until the end of the cryptoperiod of the key

6.1.2 Summary of Protection Requirements for Other Cryptographic or Related Information

Table 6 provides a summary of the protection requirements for other cryptographic information during distribution and storage. Mechanisms for providing the necessary protection are discussed in Section 6.2.

[33] The confidentiality of public ephemeral key-agreement keys may not be protected during transmission; however, the key-agreement protocols may be designed to detect unauthorized substitutions and modifications of the transmitted public ephemeral keys. In this case, the protocols form the data integrity mechanism.

Guide to Table 6:

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 (Assurance of Domain Parameter Validity) indicates the cryptographic information for which assurance **shall** be obtained as defined in [SP800-56A] and [SP800-89] and in Section 5.4 of this Recommendation. Assurance of domain-parameter validity gives confidence that domain parameters are arithmetically correct.

f. Column 6 (Period of Protection) indicates the length of time that the integrity and/or confidentiality of the cryptographic information needs to be maintained. The cryptographic information **shall** be destroyed at the end of the period of protection (see Section 8.3.4).

Table 6: Protection requirements for other cryptographic or related material

Crypto. Information Type	Security Service	Security Protection	Association Protection	Assurance of Domain Parameter Validity	Period of Protection
Domain parameters	Depends on the key assoc. with the parameters	Integrity	Usage or application; Private and public keys	Yes	From generation until no longer needed to generate keys or verify signatures
Initialization vectors	Depends on the algorithm	Integrity[34]	Protected data		From generation until no longer needed to process the protected data
Shared secrets	Support	Confidentiality; Integrity			From generation until the end of the transaction. The shared secret **shall** be destroyed at the end of the period of protection
RBG Seeds	Support	Confidentiality; Integrity	Usage or application		Used once and destroyed immediately after use

[34] IVs are not generally protected during transmission; however, the decryption system may be designed to detect or minimize the effect of unauthorized substitutions and modifications to transmitted IVs. In this case the decryption system is the data-integrity mechanism.

Other public information	Support	Integrity	Usage or application; Other authorized entities; Data processed using the nonce		From generation until no longer needed to process data using the public information
Other secret information	Support	Confidentiality; Integrity	Usage or application; Other authorized entities; Data processed using the secret information		From generation until no longer needed to process data using the secret information
Intermediate results	Support	Confidentiality; Integrity	Usage or application		From generation until no longer needed and the intermediate results are destroyed
Key-control information (e.g., IDs, purpose)	Support	Integrity	Key		From generation until the associated key is destroyed
Random number	Support	Integrity; Confidentiality (depends on usage)			From generation until no longer needed, and the random number is destroyed
Password	Source authentication; Key derivation	Integrity; Confidentiality	Usage or application; Owning entity		From generation until replaced or no longer needed to authenticate the entity or to derive keys
Audit information	Support	Integrity; Access authorization	Audited events; Key control information		From generation until no longer needed

6.2 Protection Mechanisms

During the lifetime of cryptographic information, the information is either “in transit” (e.g., is in the process of being manually distributed or distributed using automated protocols to the authorized communication participants for use by those entities), “at rest” (e.g., the information is in storage) or "in use." In all cases, the keying material **shall** be protected in accordance with Section 6.1.

For keys that are in use, the keys **shall** reside (and be used) within appropriate cryptographic modules; note that a key being in use does not preclude that key from also being simultaneously in transit and/or in storage.

While in transit or in storage, 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. In addition, the mechanisms prescribed do not, by themselves, guarantee protection. The implementation and the associated key management need to provide adequate security to prevent any feasible attack from being successful.

6.2.1 Protection Mechanisms for Cryptographic Information in Transit

Cryptographic information in transit may be keying material that is being distributed in order to obtain a cryptographic service (e.g., establish a key that will be used to provide confidentiality) (see Section 8.1.5), cryptographic information that is being backed up or archived for possible use or recovery in the future (see Sections 8.2.2 and 8.3.1), or is in the process of being recovered (see Sections 8.2.2.2, 8.3.1 and Appendix B). This may be accomplished manually (i.e., via a trusted courier), in an automated fashion (i.e., using automated communication protocols) or by some combination of manual and automated methods. For some protocols, the protections are provided by the protocol; in other cases, the protection for the keying material is provided directly to the keying material (e.g., the keying material is encrypted prior to transmission for decryption only by the receiving party). 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.

6.2.1.1 Availability

Since communications may be garbled, intentionally altered, or destroyed, the availability of cryptographic information after transit cannot be assured using cryptographic methods. However, availability can be supported by redundant or multiple channels, store and forward systems (deleting by the sender only after confirmation of receipt), error correction codes, and other non-cryptographic mechanisms.

Communication systems **should** incorporate non-cryptographic mechanisms to ensure the availability of transmitted cryptographic information after it has been successfully received, rather than relying on retransmission by the original sender for future availability

6.2.1.2 Integrity

Integrity protection involves both the prevention and detection of modifications to information. When modifications are detected, measures may be taken possible to restore the information to its unaltered form. Cryptographic mechanisms are often used to detect unauthorized modifications. 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., a CRC, MAC or digital signature) is used on the information, and the resulting code is provided to the recipient for subsequent verification. Note: A CRC may be used instead of a MAC or digital signature, since the physical protection is only intended to protect against intentional modifications.

   -OR-

   (b) The keying material is used to perform the intended cryptographic operation. If the received information does not conform to the expected format, or the data is inconsistent in the context of the application, then the keying material may have been corrupted.

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

   (a) An **approved** cryptographic integrity mechanism (e.g., a MAC or digital signature) is used on the information, and the resulting code is provided to the recipient for subsequent verification. Note that a CRC is not **approved** for this purpose. The integrity mechanism may be applied only to the cryptographic information, or may be applied to an entire message

   -OR-

   (b) The keying material is used to perform the intended cryptographic operation. If the use of the keying material produces incorrect results, or the data is inconsistent in the context of the application, then the received keying material may have been corrupted.

The response to the detection of an integrity failure will vary, depending on the specific environment. Improper error handling can allow attacks (e.g., side channel attacks). A security policy (see [SP800-57, 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 is entirely correct (e.g., the receiver cannot proceed when the information is in error), then:

a. The information **should not** be used,

b. The recipient may request that the information be resent (retransmissions **should** be limited to a predetermined maximum number of times), and

c. Information related to the incident **should** be stored in an audit log to later identify the source of the error.

6.2.1.3 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 (e.g., wrapped) using an **approved** technique that provides protection at a security strength that meets or exceeds the security strength required of the keying material.

   -OR-

   (b) The keying material is separated into key components, with each key component being generated at a security strength that meets or exceeds the security strength required of the keying material. Each key component is handled, using split knowledge procedures (see Sections 8.1.5.2.1 and 8.1.5.2.2.1), 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. Automated distribution via communication protocols: The keying material is encrypted (e.g., wrapped) using an **approved** technique that provides protection at the security strength that meets or exceeds the security strength required of the keying material.

6.2.1.4 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 6.2.3 for a discussion of the metadata associated with keys.

6.2.1.5 Association with Other Entities

The association of keying material with the appropriate entity (e.g., the entity that shares the keying material) **shall** be either specifically identified during the distribution process (e.g., using public-key certificates) or be implicitly defined by the use of the application. See Section 6.2.3 for a discussion of the metadata associated with keys.

6.2.1.6 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 6.2.3 for a discussion of the metadata associated with the other related information.

6.2.2 Protection Mechanisms for Information in Storage

Cryptographic information may be at rest in some device or storage media. This may include copies of the information that is also in transit or in use. Information-at-rest (i.e., stored information, including information contained within a cryptographic module) **shall** be protected in accordance with Section 6.1. 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 a 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.

6.2.2.1 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 8.2.1) and in backup storage (see Section 8.2.2.1). Cryptographic information that is retained after the end of a key's cryptoperiod **should** be placed in archive storage (see Section 8.3.1). This Recommendation 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 8.2.2.1, and for archive storage in Section 8.3.1.

The 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 8.2.2.2 (recovery during normal operations) and 8.3.1 (recovery from archive storage), and in Appendix B.

6.2.2.2 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 any modifications that occur (with a very high probability), and to restore the information to its original content when modifications have been detected.

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

Physical mechanisms include:

1. A validated 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 with appropriate access controls that is outside a computer system (e.g., in a safe with limited access).

Cryptographic mechanisms include:

a. An **approved** cryptographic integrity mechanism (e.g., a 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 intended cryptographic operation; this assumes that the correct result is easily determined. If the received information is incorrect, it is possible that the keying material may have been corrupted.

In order to restore the cryptographic information when an error is detected, one or more copies of the information **should** be maintained in physically separate locations (i.e., in backup or archive storage; see Sections 8.2.2.1 and 8.3.1). The integrity of each copy **should** be periodically checked.

6.2.2.3 Confidentiality

One of the following mechanisms **shall** be used to provide confidentiality for private or secret keying material in storage:

1. Encryption (or key wrapping) with an **approved** algorithm in a [FIPS140] cryptographic module; the encryption **shall** use an **approved** technique that provides protection at the security strength that meets or exceeds the security strength required of the keying material. It **shall** be no easier to recover the key-wrapping key) than it is to recover the key being encrypted (or wrapped),

   -OR-

2. Physical protection provided by a [FIPS140] cryptographic module, at a security level that is consistent with the [FIPS199] impact level associated with the data to be protected by the key (see [SP800-152]).

-OR-

3. Physical protection provided by secure storage with controlled access (e.g., a safe or protected area).

6.2.2.4 Association with Usage or Application

Cryptographic information is used with a given cryptographic mechanism (e.g., digital signatures or a key establishment scheme) or with a particular application. Protection **shall** be provided to ensure that the information is not used incorrectly (e.g., not only must the usage or application be associated with the keying material, but the integrity of this association must be maintained). This protection can be provided by separating the cryptographic information from that of other mechanisms or applications, or by the use of appropriate metadata associated with the information. Section 6.2.3 addresses the metadata associated with cryptographic information.

6.2.2.5 Association with the Other Entities

Some cryptographic information needs to be correctly associated with another entity (e.g., the key source), and the integrity of this association **shall** be maintained. For example, a symmetric (secret) key used for the encryption of information, or the computation of a MAC needs to be associated with the other entity(ies) that share(s) the key. Public keys need to be correctly associated (e.g., cryptographically bound) with the owner of the key pair (e.g., using public-key certificates).

The cryptographic information **shall** retain its association during storage by separating the information by "entity" or application, or by using appropriate metadata for the information. Section 6.2.3 addresses the metadata used for cryptographic information.

6.2.2.6 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 6.2.1.6).

Storing the information together or providing some linkage or pointer between the information accomplishes the association. Often, the linkage between a key and the information it protects is accomplished by providing an identifier for a key, storing the identifier with the key in the key's metadata, and storing the key's identifier with the protected information. The association **shall** be maintained for as long as the protected information needs to be processed.

Section 6.2.3 addresses the use of metadata for cryptographic information.

6.2.3 Metadata Associated with Cryptographic Information

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

6.2.3.1 Metadata for Keys

Metadata is used to provide information about the key, including its parameters, or the intended use of a key, and as such, contains the key's control information. Different applications may require different metadata elements for the same key type, and different metadata elements may be required for different key types. It is the responsibility of an implementer to select suitable metadata elements for keys. When metadata is used, the metadata **should** accompany a key (i.e., the metadata is typically stored or transmitted with a key). Some examples of metadata elements are:

1. Key identifier;
2. Information identifying associated keys (e.g., the association between a public and private key);
3. Identity of the key's owner or the sharing entity(ies);
4. Cryptoperiod (e.g., the start date and end date);
5. Key type (e.g., a signing private key, encryption key, or master key);
6. Application (e.g., purchasing, email);
7. Sensitivity of the information protected by the key;
8. Counter[35];
9. Domain parameters (e.g., the domain parameters used by DSA or ECDSA, or a pointer to them);
10. Key state (e.g., pre-activation, active, destroyed);
11. Key status/history (e.g., distributed, revoked (with the revocation reason));
12. Key-wrapping key identifier and the algorithm used for wrapping;
13. Integrity-protection mechanism (e.g., the key and algorithm used to provide cryptographic protection, and the protection code (e.g., MAC, digital signature)); and
14. Other information (e.g., the length of the key, any protection requirements, who has access rights to the key, additional conditions for use).

[SP800-152] provides additional information about the use of metadata, including guidance about protecting its integrity and association with the related key.

6.2.3.2 Metadata for Related Cryptographic Information

Cryptographic information other than keying material may need metadata to "point to" the keying material that was used to provide the cryptographic protection for the information. The metadata may also contain other related cryptographic information. When metadata is used, the metadata **should** accompany the information (i.e., the metadata 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. The source of the information (e.g., the entity that sent the information);
3. The application for using the key (e.g., purchasing, email);
4. Other associated cryptographic information (e.g., a key, MAC or hash value); and
5. Any other information (e.g., who has access rights).

[35] Used to detect the playback of a previously transmitted key package.

7 Key States and Transitions

[Note to the reviewer: Please review this section carefully to see if it makes sense and is clear.]

A key may pass through several states between its generation and its destruction. Figure 3 depicts an example of the key states that a key could assume and the transitions among them.

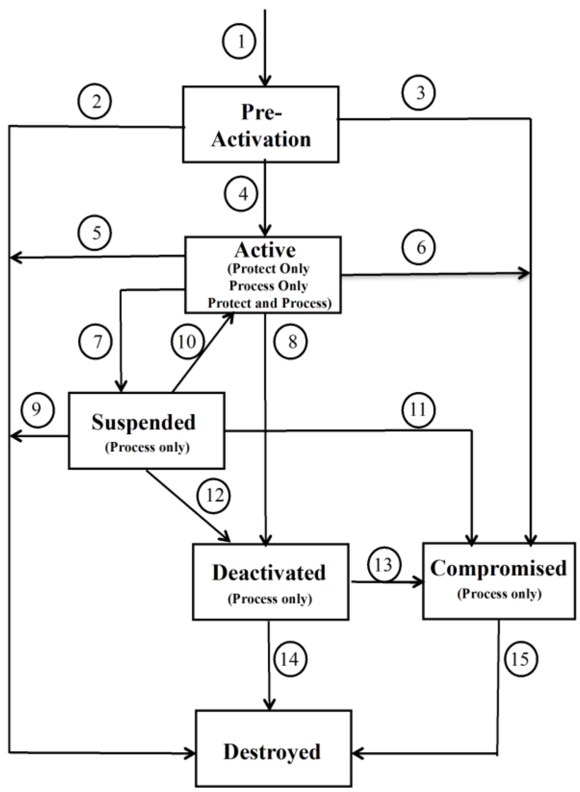

Figure 3: Key states and transitions.

A key is used differently, depending upon its state in the key's lifecycle. Key states are defined from a system point-of-view, as opposed to the point-of-view of a single cryptographic module. The following sections discuss the states that an operational or backed-up key may assume, along with transitions to other states, as shown in Figure 3. Additional states may be applicable for some systems, and some of the identified states may not be needed for other systems (e.g., if keys are to be activated immediately after generation, the pre-activation state may not be needed, or a decision could be made that the suspended state will not be used).

Transitioning between states often requires recording the event. Suitable places for such recordings are audit logs and the key's metadata (see Section 6.2.3.1). [SP800-152] also discusses the logging of these events.

7.1 Pre-activation State

The key has been generated, but has not been authorized for use. In this state, the key may only be used to perform proof-of-possession or key confirmation. Other than for proof-of-possession (Section 8.1.5.1.1.2) or key-confirmation (Section 4.2.5.5) purposes, a key **shall not** be used to apply cryptographic protection to information (e.g., encrypt or sign information to be transmitted or stored) or to process cryptographically protected information (e.g., decrypt ciphertext or verify a digital signature) while in this state.

Transition 1: A key enters the pre-activation state immediately upon generation.

Transition 2: If a key is in the pre-activation state, and it has been determined that the key will not be needed in the future, the key **shall** transition directly from the pre-activation state to the destroyed state.

In the case of asymmetric keys, both keys of the key pair **shall** transition to the destroyed state.

The date and time of the transition **shall** be recorded.

Transition 3: When a key is in the pre-activation state, and the integrity of the key or the confidentiality of a key requiring confidentiality protection becomes suspect, then the key **shall** transition from the pre-activation state to the compromised state.

In the case of asymmetric keys, both keys of the key pair **shall** transition to the compromised state.

The date and time of the transition **shall** be recorded. If the key is known by multiple entities, a revocation notice **shall** be generated.

Transition 4: Keys **shall** transition from the pre-activation state to the active state when the key becomes available for use. This transition may occur upon reaching an activation date or may occur because of an external event. In the case where keys are generated for immediate use, this transition occurs immediately after entering the pre-activation state.

For certified asymmetric keys, both keys of the key pair become active upon the *notBefore* date in the first certificate issued for the public key of the key pair.

The date and time of the transition **should** be recorded.

This transition marks the beginning of the cryptoperiod of a symmetric key or both keys of an asymmetric key pair (see Section 5.3).

7.2 Active State

The key may be used to cryptographically protect information (e.g., encrypt plaintext or generate a digital signature), to cryptographically process previously protected information (e.g., decrypt ciphertext or verify a digital signature) or both. When a key is active, it may be designated for protection only, processing only, or both protection and processing, depending on its type. For example, private signature keys and public key-transport keys are implicitly designated for only applying protection; public signature-verification keys and private key-transport keys are designated for processing only. A symmetric data-encryption key may be used to encrypt data during its originator-usage period and decrypt the encrypted data during its recipient-usage period (see Section 5.3.5).

Transition 5: Several key types transition directly from the active state to the destroyed state if no compromise has been determined and either the key's cryptoperiod has been reached or the key has been replaced.

Private signature keys and private authentication keys **shall** transition to the destroyed state at the end of their respective originator-usage periods (e.g., when the *notAfter* dates are reached on the last certificate issued for the corresponding public keys). Note that the corresponding public keys transition to the deactivated state at this time; see transition 8.

A symmetric RBG key **shall** transition to the destroyed state when replaced by a new key or when the RBG will no longer be used.

Symmetric master keys and symmetric authorization keys **shall** transition to the destroyed state at the end of their respective originator-usage periods[36].

Private ephemeral key-agreement keys **shall** transition to the destroyed state immediately after use (see [SP800-56A]). The corresponding public ephemeral key-agreement keys **should** transition to the destroyed state when the corresponding private keys are destroyed[37].

A private authorization key **shall** transition to the destroyed state at the end of its cryptoperiod (e,g., when the *notAfter* dates is reached on the last certificate issued for the corresponding public key). A public authorization key **should** transition to the destroyed state when the corresponding private key is destroyed[38].

The date and time of the transition **shall** be recorded.

Transition 6: A key or key pair **shall** transition from the active state to the compromised state when the integrity of the key or the confidentiality of a key requiring confidentiality protection becomes suspect. In this case, the key or key pair **shall** be revoked.

In the case of asymmetric key pairs, the compromise pertains explicitly to the private key of the key pair, but both keys **shall** transition to the compromised state. For example, when a private signature key or private key-transport key is either compromised or suspected of being compromised, the corresponding public key also needs to transition to the compromised state.

---

[36] Recall that the recipient-usage periods of symmetric key-agreement keys and symmetric authorization keys are the same as their originator-usage periods (see Section 5.6).

[37] Recall that the cryptoperiods of the private and public authorization keys are the same (see Section 5.6).

[38] Recall that the cryptoperiods of the private and public authorization keys are the same (see Section 5.6).

The date and time of the transition **shall** be recorded. If the key is known by multiple entities, a revocation notice **shall** be generated.

Transaction 7: A key or key pair **shall** transition from the active state to the suspended state if, for some reason, the key is not to be used for a period of time. For example, a key may be suspended because the entity associated with the key is on a leave of absence.

In the case of asymmetric keys, both keys of the key pair **shall** transition to the suspended state at the same time.

Symmetric RBG keys **shall** transition to the compromised state and be replaced, rather than suspended.

The date, time and reason for the suspension **shall** be recorded. If the key or key pair is known by multiple entities, a notification indicating the suspension and reason **shall** be generated.

Transition 8: A key or key pair in the active state **shall** transition to the deactivated state when it is no longer to be used to apply cryptographic protection to data. The transition to the deactivated state may be because a symmetric key was replaced (see Section 8.2.3), because the end of the originator-usage cryptoperiod has been reached (see Sections 5.3.4 and 5.3.5) or because the key or key pair was revoked for reasons other than a compromise (e.g., the key's owner is no longer authorized to use the key).

Symmetric authentication keys, symmetric data encryption/decryption keys, symmetric key-agreement keys and key wrapping keys transition to the deactivated state at the end of the key's originator-usage period.

Public signature verification keys, public authentication keys, and private/public static key-agreement key pairs, transition to the deactivated state at the end of the originator-usage period for the corresponding private key (e.g., when the *notAfter* date is reached on the last certificate issued for the public key). Public ephemeral key-agreement keys and public authorization keys transition to the deactivated state if they have not been destroyed when the corresponding private keys were destroyed (see transition 5).

A private and public key-transport key pair transitions to the deactivated state when the *notAfter* date is reached on the last certificate issued for the public key.

The date and time of the transition **should** be recorded.

7.3 Suspended State

The use of a key or key pair may be suspended for several possible reasons; in the case of asymmetric key pairs, both the public and private keys **shall** be suspended at the same time. One reason for a suspension might be a possible key compromise, and the suspension has been issued to allow time to investigate the situation. Another reason might be that the entity that owns a digital signature key pair is not available (e.g., is on an extended leave of absence); signatures purportedly signed during the suspension time would be invalid.

A suspended key or key pair may be restored to an active state at a later time or may be deactivated or destroyed, or may transition to the compromised state.

A suspended key **shall not** be used to apply cryptographic protection (e.g., encrypt plaintext or generate a digital signature). However, a suspended key could be used to process information that was protected prior to the suspension (e.g., decrypt ciphertext or verify a digital signature), but the recipient must accept the risk in doing so (e.g., the recipient must understand the reason and implications of the suspension). For example, if the reason for the suspension is because of a suspected compromise, it may not be prudent to verify signatures using the public key unless the key pair is subsequently reactivated. Information for which protection is known to be applied during the suspension period **shall not** be processed until leaving the suspended state, at which time its processing depends on the new state.

Transition 9: Several key types transition from the suspended state to the destroyed state if no compromise has been determined.

Private signature keys and private authentication keys in the suspended state **shall** transition to the destroyed state at the end of their originator-usage periods (e.g., when the *notAfter* dates are reached on the last certificate issued for the corresponding public keys). Note that the corresponding public keys transition to the deactivated state at this time (see transition 12).

Symmetric master keys and symmetric authorization keys in the suspended state **shall** transition to the destroyed state at the end of their originator-usage periods[39].

Private authorization keys in the suspended state **shall** transition to the destroyed state at the end of their originator-usage periods (i.e., when the *notAfter* dates are reached on the last certificate issued for the corresponding public keys). Public authorization keys **should** transition to the destroyed state when the corresponding private keys are destroyed[40].

The date and time of the transition **shall** be recorded.

Transition 10: A key or key pair in the suspended state **shall** transition to the active state when the reason for the suspension no longer exists, and the end of the originator-usage period has not been reached.

In the case of symmetric keys, the transition needs to be made before the end of the key's originator-usage period.

For asymmetric keys, the transition needs to be made, for example, before the *notAfter* date on the last certificate issued for the public key. In this case, both the private and public key **shall** transition at the same time.

The date and time of the transition **should** be recorded.

[39] Recall that the recipient-usage periods of symmetric key-agreement keys and symmetric authorization keys are the same as their originator-usage periods (see Section 5.3.6).

[40] Recall that the cryptoperiods of the private and public authorization keys are the same (see Section 5.6).

Transition 11: A key or key pair in the suspended state **shall** transition to the compromised state when the integrity of the key or the confidentiality of a key requiring confidentiality protection becomes suspect. In this case, the key or key pair **shall** be revoked.

In the case of asymmetric key pairs, both the public and private keys **shall** be transition at the same time.

The date and time of the transition **shall** be recorded. If the key is known by multiple entities, a revocation notice **shall** be generated.

Transition 12: Several key types transition from the suspended state to the deactivated state if no compromise has been determined and the suspension is no longer required.

Symmetric authentication keys, symmetric data encryption/decryption keys, and symmetric key-wrapping keys **shall** transition to the deactivated state when the ends of their originator-usage periods have been reached.

Public signature verification keys, public authentication keys, and private/public static key-agreement key pairs[41] transition to the deactivated state at the end of the private key's originator-usage period (e.g., when the *notAfter* date is reached on the last certificate issued for the public key). Public ephemeral key-agreement keys and public authorization keys transition to the deactivated state if they have not been destroyed when the corresponding private keys were destroyed (see transition 9).

A private/public key-transport key pair transitions to the deactivated state at the end of the key pair's cryptoperiod (e.g., when the *notAfter* date is reached on the last certificate issued for the public key).

The date and time of the transition **should** be recorded.

7.4 Deactivated State

Keys in the deactivated state **shall not** be used to apply cryptographic protection, but in some cases, may be used to process cryptographically protected information. If the key has been revoked (i.e., for reasons other than a compromise), then the key may continue to be used for processing. Note that keys retrieved from an archive can be considered to be in the deactivated state unless compromised.

- Public signature verification keys may be used to verify the digital signatures generated before the end of the private key's originator-usage period (e.g., before the *notAfter* date in the last certificate for the public key).
- Symmetric authentication keys, symmetric data encryption keys and symmetric key-wrapping keys may be used to process cryptographically protected information until the

[41] In the case of public ephemeral key-agreement keys, the cryptoperiod ends at the same time as that of the corresponding private ephemeral key-agreement key (which transitioned to the destroyed state after use (see transition 5), However, there is no actual requirement to destroy the public key immediately, so it is listed here as transitioning to the deactivated state, rather than the destroyed state. However, transitioning directly to the destroyed state would also be acceptable.

end of the recipient-usage period is reached, provided that the protection was applied during the key's originator-usage period.

- Public authentication keys may be used to authenticate processes performed before the end of the corresponding private key's originator-usage period (e.g., before the *notAfter* date in the last certificate for the public key).
- Private key-transport keys may be used to decrypt keys that were encrypted using the corresponding public key before the end of the public key's originator-usage period (e.g., before the *notAfter* date in the last certificate for the public key).
- Symmetric key-agreement keys may be used to determine the agreed-upon key, assuming that sufficient information is available.
- Private/public static key-agreement keys may be used to regenerate agreed-upon keys that were created before the end of the key pair's cryptoperiod (e.g., before the *notAfter* date in the last certificate for the public key, assuming that sufficient information is available for the key-agreement scheme used).
- Public ephemeral key-agreement keys may be used to regenerate agreed-upon keys (assuming that sufficient information is available for the key-agreement scheme used).
- Public authorization keys **shall not** be used.

Keys in the deactivated state may transition to either the compromised or destroyed state at some point in time.

Transition 13: A key **shall** transition from the deactivated state to the compromised state when the integrity of a key or the confidentiality of a key requiring confidentiality protection becomes suspect. In this case, the key or key pair **shall** be revoked.

The date, time and reason for the transition **shall** be recorded. If the key is known by multiple entities, a revocation notice **shall** be generated.

Transition 14: A key in the deactivated state **should** transition to the destroyed state as soon as it is no longer needed.

The date, time and reason for the transition **shall** be recorded.

Note that keys retrieved from an archive may be in the deactivated state.

7.5 Compromised State

Generally, keys are compromised when they are released to or determined by an unauthorized entity. A compromised key **shall not** be used to apply cryptographic protection to information. However, in some cases, a compromised key or a public key that corresponds to a compromised private key of a key pair may be used to process cryptographically protected information. For example, a signature may be verified to determine the integrity of signed data if its signature has been physically protected since a time before the compromise occurred. This processing **shall** be done only under very highly controlled conditions, where the users of the information are fully aware of the possible consequences.

Note that keys retrieved from an archive may be in the compromised state.

Transition 15: A compromised key **should** transition to the destroyed state when its use will no longer be allowed or needed.

The date and time of the transition **shall** be recorded.

7.6 Destroyed State

The key has been destroyed as specified in Section 8.3.4. Even though the key no longer exists when in this state, certain key metadata (e.g., key state transition history, key name, type, and cryptoperiod) may be retained (see Section 8.4).

It is possible that a compromise of the destroyed key could be determined after the key has been destroyed. In this case, the compromise **should** be recorded.

8 Key-Management Phases and Functions

The cryptographic key-management lifecycle can be divided into four phases. During each phase, the keys are in certain specific key states as discussed in Section 7. In addition, within each phase, certain key-management functions are typically performed. These functions are necessary for the management of the keys and their associated metadata.

Key-management information is called metadata. The metadata 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. Metadata is used by applications to select the appropriate cryptographic key(s) for a particular service. While the metadata does not appear in cryptographic algorithms, it is crucial to the implementation of applications and application protocols.

The four phases of key management are specified below.

1. **Pre-operational phase:** The keying material is not yet available for normal cryptographic operations. Keys may not yet be generated, or are 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, suspended or deactivated state. Keys in the active state may be designated as protect only, process only, or protect and process; keys in the suspended or deactivated state can be used for processing only.
3. **Post-operational phase**: The keying material is no longer in normal use, but access to the keying material is possible, and the keying material may be used for processing only in certain circumstances. Keys are in the deactivated or compromised states. Keys in the post-operational phase may be in an archive (see Section 8.3.1) when not processing data.
4. **Destroyed phase:** Keys are no longer available. Records of their existence may or may not have been deleted. Keys are in the destroyed states. Although the keys themselves are destroyed, the key metadata (e.g., key name, type, cryptoperiod, and usage period) may be retained (see Section 8.4).

A flow diagram for the key management phases is presented in Figure 4. Seven phase transitions are identified in the diagram. A key **shall not** be able to transfer back to any previous phase.

Transition 1: A key is in the pre-operational phase upon generation (pre-activation state).

Transition 2: If keys are produced, but never used, they may be destroyed by transitioning from the pre-operational phase directly to the destroyed phase.

Transition 3: When a key in the pre-operational phase is compromised, it transitions to the post-operational phase (compromised state).

Transition 4: After the required key metadata has been established, keying material has been generated, and the metadata is associated with the key during the pre-operational phase, the key is ready to be used by applications and transitions to the operational phase at the appropriate time.

Transition 5: When a key in the operational phase is compromised, it transitions to the post-operational phase (compromised state).

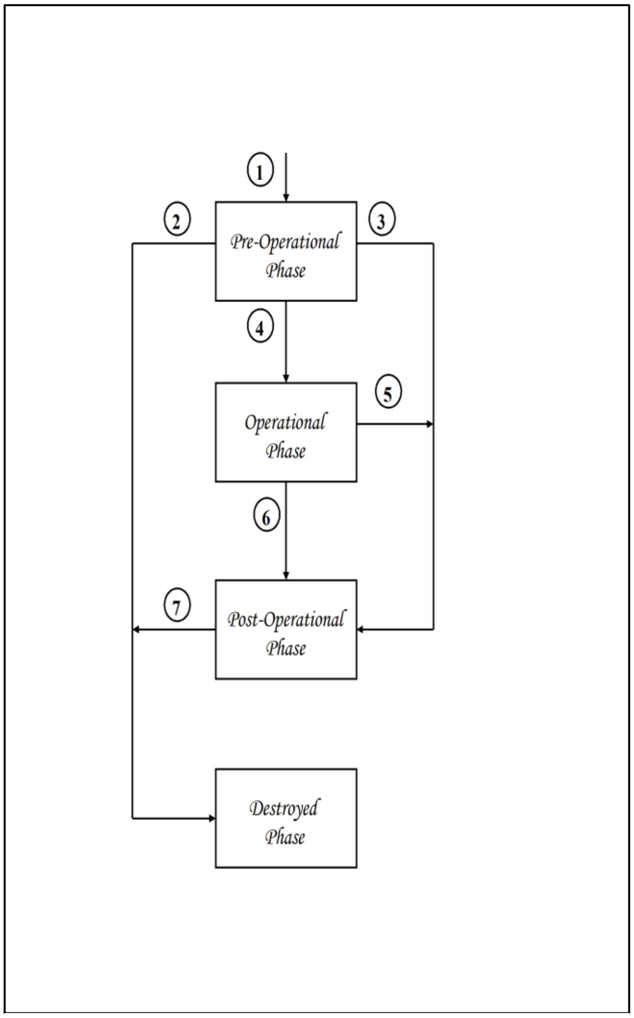

Figure 4: Key management phases

Transition 6: 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.

Transition 7: Some applications will require that access be preserved for a period of time, and 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 destroyed phase.

The combination of key states and key phases is illustrated in Figure 5. The pre-operational and destroyed phases contain only one state each, while the operational and post-operational phase have two states.

The following subsections discuss the functions that are performed in each phase of key management. 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 functions of the post-operational phase if keys are immediately destroyed when they are no longer used to apply cryptographic protection or are compromised. In this case, keys would move from the operational phase directly to the destroyed phase.

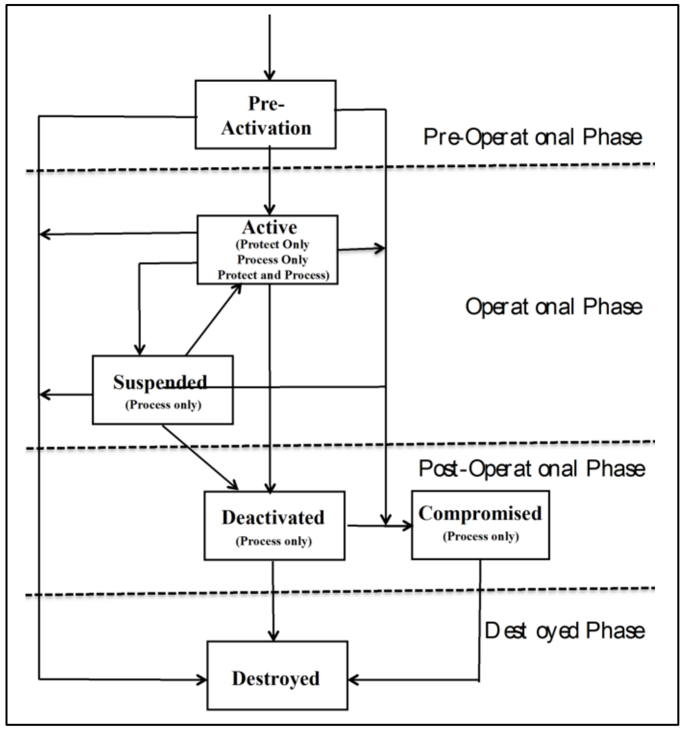

Figure 5: Key management states and phases

8.1 Pre-operational Phase

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

8.1.1 User Registration Function

During user registration, an entity interacts with a registration authority to become an authorized member of a security domain. In this phase, a user identifier 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 Sections 8.1.5 and 8.1.6). The entity may also establish various information during the registration function, such as email addresses, or role and authorization information. As with identity information, this information 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. Human entities are usually required to provide credentials (e.g., an identification card or birth certificate), while system entities are vouched for by those individuals responsible for system operation. The strength (or weakness) of a security infrastructure will often depend upon the identification process.

User and key registration (see Section 8.1.6) 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's identity during the key registration step. If performed in concert, the user establishes an identity and performs key registration in the same process, so the secret value is not required.

8.1.2 System Initialization Function

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., recognized certificate policies).

8.1.3 User Initialization Function

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 8.1.4) 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.

8.1.4 Keying-Material Installation Function

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, cryptographic module, or device using a variety of techniques. Keying material is installed during initial set up, when new keying material is added to the existing keying material, and when existing keying material is replaced (e.g., via re-keying or key derivation − see Sections 8.2.3 and 8.2.4).

The process for the initial installation of keying material (e.g., by manual entry, electronic key loader, or a vendor during manufacture) **shall** include the protection of the keying material during entry into a software/hardware/system/application/device/cryptographic module, taking into account the requirements of [FIPS140] and its differing requirements for the different levels of protection, and include any additional procedures that may be required.

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.

8.1.5 Key Establishment Function

Key establishment involves the generation and distribution, or the agreement of keying material for communication between entities. All keys **shall** be generated within a FIPS140-validated cryptographic module or obtained from another source approved by the U.S. Government for the protection of national security information. During the key-establishment process, some of the keying material may be in transit (i.e., the keying material is being manually distributed or is being distributed using automated protocols). Other keying material may be retained locally. In either case, the keying material **shall** be protected in accordance with Section 6.

An entity may be an individual (person), organization, device or process. When keying material is generated by an entity for its own use, one or more of the appropriate protection mechanisms for stored information in Section 6.2.2 **shall** be used.

Keying material that is distributed between entities, or among an entity and its sub-entities (e.g., various individuals, devices or processes within an organization), **shall** be protected during distribution using one or more of the appropriate protection mechanisms specified in Section 6.2.1. 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) or keying material that is received and subsequently stored **shall** be protected using one or more of the appropriate protection mechanisms specified in Section 6.2.2.

[SP800-133] discusses the generation of keying material.

8.1.5.1 Generation and Distribution of Asymmetric Key Pairs

Key pairs **shall** be generated in accordance with the mathematical specifications of the appropriate **approved** FIPS or NIST Recommendation.

A static key pair **shall** be 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 8.1.5.1.3, or by the user and facility in a cooperative process. When generated by the entity that owns the key pair, a signing private key **shall not** be distributed to other entities. In the case of a public signature-verification 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 the support for non-repudiation. However, when the owner is an organization, it is acceptable to distribute the keying material to the organization's sub-entities (e.g., employees or devices); in this case, the organization is the true owner, and the sub-entities represent the owner.

Ephemeral keys are often used for key establishment (see [SP800-56A]). They are generated for each new key-establishment transaction (e.g., unique to each message or session).

The generated key pairs **shall** be protected in accordance with Section 6.1.1.

8.1.5.1.1 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 the public key that the true owner of the key is known (i.e., the claimed owner is the actual owner); this requirement may be disregarded if anonymity is acceptable. However, the strength of the overall architecture and trust in the validity of the protected data depends, in large part, on the assurance of the public-key owner's identity.

In addition, the distribution of the public key **shall** provide assurance to the receiver that:

1. The purpose/usage of the key is known (e.g., for RSA digital signatures or elliptic-curve key agreement),
2. Any parameters associated with the public key are known (e.g., domain parameters),
3. The public key is valid (e.g., the public key satisfies the required arithmetical properties), and
4. The owner actually possesses the corresponding private key.

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

The public key of a trusted Certification Authority is the foundation for all PKI-based security services; the trusted CA is considered to be a trust anchor. The trust anchor's public key is not a secret, but the *authenticity* of that public key is the crucial assumption for PKI. Trust anchor public keys 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 anchor public keys are frequently distributed as "self-signed" X.509 certificates, that is, certificates that are signed by the private key corresponding to the public key in the certificate. Note that, while this document refers to a trusted CA as the "trust anchor" and its certificate as the "trust anchor certificate," many other documents use the term "trust anchor" to refer to both the trusted CA and the CA's certificate.

Trust anchor certificates 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 replacement of the user's list of trust anchor certificates. Operating systems are often shipped with "code signing" trust anchor certificates. The user relies upon the authenticity of the software distribution mechanism to ensure that only valid trust anchor certificates are installed during installation or replacement. However, in some cases other applications may install trust anchor certificates in web browsers.

Trust anchor certificates 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 TLS protocol to authenticate the web server and provide confidentiality. Users who visit a "secure" website that has 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. **Relying users should be cautious about accepting certificates from unknown Certification Authorities so that they do not, in effect, inadvertently add new permanent trust anchor certificates that are really not trustworthy**.

Warning: Roaming users **should** be aware that they are implicitly trusting all software on the host systems that they use. They should have concerns about trust anchor certificates used by web browsers when they use systems in kiosks, libraries, Internet cafes, or hotels, as well as systems provided by conference organizers to access "secure websites." The user has had no control over the trust anchor certificates installed in the host system, and therefore the user is relying upon the host systems to have made good, sensible decisions about which trust anchor certificates are allowed; relying parties are not participants in trust anchor certificate selection when the trust anchor certificates are pre-installed prior to software distribution, and may have had no part in decisions about which trust anchor certificates are installed thereafter. The user should be aware that he is trusting the software distribution mechanism to avoid the installation of malicious code. Extending this trust to cover trust anchor certificates for a given application may be reasonable, and allows the relying party to obtain trust anchor certificates without any additional procedures.

When a user registers keys with an infrastructure, additional mechanisms are usually available. The user interacts securely with the infrastructure to register its keys (e.g., to obtain certificates), and these interactions may be extended to provide trust anchor information in the form of a trust anchor certificate. This allows the user to establish trust anchor certificates 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 a trust anchor certificate **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 trust anchor's public key, associated parameters, key use, and assurance of possession are conveyed as a self-signed X.509 public-key certificate. In this case, the certificate has been digitally signed

by the private key that corresponds to the public key within the certificate. While the parameters and assurance of possession may be conveyed in the self-signed certificate, the identity associated with the trust anchor certificate and other information cannot be verified from the self-signed certificate itself (see item 2 below).

2. The trusted process used to convey a requesting entity's public key and assurances to the RA or CA **shall** also be used to protect the trust anchor's certificate that is conveyed to the requesting entity. In cases where the requesting entity appears in person, the trust anchor's certificate may be provided at that time. If a secret value has been established during user registration (see Section 8.1.1), the trust anchor's certificate may be supplied, along with the requesting entity's certificate.

8.1.5.1.1.2 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 **shall** obtain the assurances listed in Section 8.1.5.1.1 from the owner of the key or an authorized representative (e.g., the firewall administrator), including the owner's identity.

In general, the owner of the key is identified in terms of an identifier established during user registration (see Section 8.1.1). The key owner identifies the appropriate uses for the key, along with any required parameters. In cases where anonymous ownership of the public key is acceptable, the owner or the registration authority determines a pseudonym to be used as the identifier. The identifier **shall** be unique for the naming authority[42].

Proof of Possession (POP) is a mechanism that is commonly used by a CA to obtain assurance of private-key possession during key registration. In this case, the proof **shall** be provided by the reputed owner of the key pair. Without assurance of possession, it would be possible for the CA to bind the public key to the wrong entity.

The (reputed) owner **should** provide POP by performing operations with the private key that satisfy the indicated key use. For example, if a key pair is intended for RSA digital signature generation, the CA may provide information to be signed using the owner's private key. If the owner can correctly verify the signature using the corresponding public key, then the owner has established POP. However, when a key pair is intended to support key establishment (i.e., either key agreement or key transport), POP may also be afforded by using the private key to digitally sign the certificate request (although this is not the preferred method). The private key-establishment key (i.e., the private key-agreement or private key-transport key) **shall not** be used to perform signature operations after certificate issuance.

As with user registration, the strength of the security infrastructure depends upon the methods used for distributing the key to an 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 and the information identified in Section 8.1.5.1.1 are provided in person by the public-key owner in person, or by an authorized representative of the public-key owner.

[42] The naming authority is the entity responsible for the allocation and distribution of domain names, ensuring that the names are unique within the domain. A naming authority is often restricted to a particular level of domains, such as .com, ,net or .edu.

2. The identity of the public-key owner or an authorized representative of the public-key owner (i.e., a person, organization, device or process) is established at the RA or CA in person 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 or authorized representative as a secret value. The public key and the information identified in Section 8.1.5.1.1 are provided to the RA or CA using a communication protocol protected by the secret value. The secret value **should** be destroyed by the key owner as specified in Section 8.3.4 upon receiving confirmation that the certificate has been successfully generated. The RA or CA may maintain this secret value for auditing purposes, but the RA or CA **should not** accept further use of the secret value to prove identity.

   When a specific list of public-key owners are pre-authorized to register keys, identifiers may be assigned without the owners being present. In this case, it is critical to protect the secret values from disclosure, and the procedures **shall** demonstrate that the chain of custody was maintained. The lifetime of the secret values **should** be limited, but **shall** allow for the public-key owner to appear at the RA or CA, to generate his keys, and to 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 for the secret value is probably reasonable.

   When public-key owners are not pre-authorized, the RA or CA **shall** determine 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 his 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 digital-signature key pair. For example, the request for a new public-key certificate is signed by the owner of the new public key to be certified. The private signature key used to sign the request **should** correspond to a public signature-verification key that is certified by the same CA that will certify the new public key. The request contains the new public key and any key-related information (e.g., the key use and the key's parameters). In addition, the CA **shall** obtain assurance of public-key validity and assurance that the owner possesses the corresponding private key.

4. The public key, key use, parameters, validity assurance information, and assurance of possession 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 containing the requested certificate). 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 a trusted process (e.g., registered mail sent via the U.S. Postal Service). The trusted process assures that the identity of the requestor is verified 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 subsequently delivers the certificate to the owner.

The unique, unpredictable information **should** be destroyed by the key owner as specified in Section 8.3.4 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 RA and CA, in combination, **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 certificate policy or certification practice statement). After generation, the certificate is distributed manually or using automated protocols to the RA, the public-key owner, or a certificate repository (i.e., a directory) in accordance with the CA's certification practice statement.

8.1.5.1.1.3 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 mandatory assurances listed in Section 8.1.5.1.1 **shall** be provided to the recipient prior to the use of the public key operationally.

2. Manual (e.g., in a face-to-face transfer or by receipted mail) or automated distribution of a public-key certificate by the public-key owner, the CA, or a certificate repository (i.e., a directory). The mandatory assurances listed in Section 8.1.5.1.1 that are not provided by the CA (e.g., public-key validation) **shall** be provided to or performed by the receiver of the public key prior to the use of the key operationally.

3. Automated distribution of a public key (e.g., using a communication protocol with authentication and content integrity). The mandatory assurances listed in Section 8.1.5.1.1 **shall** be provided to the receiving entity prior to the use of the public key operationally.

8.1.5.1.2 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 + key confirmation + any associated negotiation + local processing) **should** provide a recipient with the assurances listed in Section 8.1.5.1.1. The recipient of an ephemeral public key **shall** obtain assurance of validity of that key as specified in [SP800-56A] prior to using that key for subsequent steps in the key-agreement process.

8.1.5.1.3 Distribution of Centrally Generated Key Pairs

When a static key pair is centrally generated, the key pair **shall** be generated within a FIPS140-validated cryptographic module or obtained from another source approved by the U.S. government for protecting national security information 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 support for non-repudiation for those individual subscribers; therefore, when support for 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 support for non-repudiation may be 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** include an authentication of the recipient's identity as established during user registration (see Section 8.1.1).

The key pair may be distributed to the intended owner using an appropriate manual method (e.g., courier, mail or other method specified by the key-generation facility) or secure automated method (e.g., a secure communication protocol). The private key **shall** be distributed in the same manner as a symmetric key (see Section 8.1.5.2.2). During the distribution process, each key of the key pair **shall** be provided with the appropriate protections for that key (see Section 6.1).

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

Upon receipt of the key pair, the owner **shall** obtain assurance of the validity of the public key (see [SP800-56A], [SP800-56B] and [SP800-89]). The owner **shall** obtain assurance that the public and private keys of the key pair are correctly associated (i.e., check that they are a consistent pair, for example, by checking that a key encrypted under a public key-transport key can be decrypted by the private key-transport key).

8.1.5.2 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 6.

Symmetric keys **shall** be either:

1. Generated and subsequently distributed (see Sections 8.1.5.2.1 and 8.1.5.2.2) either manually (see Section 8.1.5.2.2.1), using a public key-transport mechanism (see Section 8.1.5.2.2.2), or using a previously distributed or agreed-upon key wrapping key (see Section 8.1.5.2.2.2),
2. Established using a key-agreement scheme (i.e., the generation and distribution are accomplished with one process) (see Section 8.1.5.2.3), or
3. Derived from a master key (see Section 8.2.4).

8.1.5.2.1 Key Generation

Symmetric keys determined by key generation methods **shall** be either generated by an **approved** method (e.g., using an **approved** random number generator), or derived from a master key (see Section 8.2.4) using an **approved** key-derivation function (see [SP800-108]). Also, see [SP800-133].

When split-knowledge procedures are used, the key **shall** exist outside of a [FIPS140] cryptographic module as multiple key components. The keying material may be created within a cryptographic module and then split into components for export from the module, or may be created as separate components. Each key component **shall** provide no knowledge of the key value (e.g., each key component must appear to be generated randomly). If knowledge of *k* components is required to construct the original key, then knowledge of any *k*-1 key components **shall** provide no information about the original key other than, possibly, its length. Note: A suitable combination function is not provided by simple concatenation; e.g., it is not acceptable to form a 128-bit key by concatenating two 64-bit key components.

All keys and key components **shall** be generated within a FIPS 140-validated cryptographic module or obtained from another source approved by the U.S. Government for the protection of national security information.

8.1.5.2.2 Key Distribution

Keys generated in accordance with Section 8.1.5.2.1 as key-wrapping keys (i.e., key-encrypting keys), 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 automated key-transport protocol (automated key transport).

Keys used only for the storage of information (i.e., data or keying material) **shall not** be distributed except for backup or to other authorized entities that may require access to the stored information protected by the keys.

8.1.5.2.2.1 Manual Key Distribution

Keys distributed manually (i.e., by other than an automated key-transport protocol) **shall** be protected throughout the distribution process. During manual distribution, secret or private keys **shall** either be wrapped (i.e., 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 that:

1. The distribution of the keys is from an authorized source,
2. Any entity distributing plaintext 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 6, and
4. The keys are received by the authorized recipient.

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

When using split knowledge procedures, each key component **shall** be either encrypted or distributed separately to each 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 9.1 and 9.2) **should** be used.

8.1.5.2.2.2 Automated Key Distribution/Key Transport/Key Wrapping

Automated key distribution, also known as key transport or key wrapping, is used to distribute keys via a communication channel (e.g., the Internet or a satellite transmission). This requires the prior distribution of a key-wrapping key (i.e., a key-encryption key) or a public key-transport key as follows:

1. A key-wrapping key **shall** be generated and distributed in accordance with Sections 8.1.5.2.1 and 8.1.5.2.2, or established using a key-agreement scheme as defined in Section 8.1.5.2.3.
2. A public key-transport key **shall** be generated and distributed as specified in Section 8.1.5.1.

Only **approved** key-wrapping or public key-transport schemes **shall** be used. The **approved** schemes provide assurance that:

a. For symmetric key-wrapping schemes: The key-wrapping key and the distributed key are not disclosed or modified. **Approved** key-wrapping algorithms are provided in [SP800-38F]. Note that in this case, key encryption alone, as discussed in Section 4.2.5.4, does not provide protection against modification; an additional integrity mechanism must be used (e.g., by using an authenticated encryption mode).

b. For asymmetric key-transport schemes: The private key-transport key and the distributed key are not disclosed or modified, and correct association between the private and public key-transport keys is maintained. **Approved** key-transport schemes using asymmetric techniques are provided in [SP800-56A] and [SP800-56B].

c. The keys are protected in accordance with Section 6.

In addition, the **approved** schemes, together with the associated key-establishment protocol, **should** provide the following assurances:

d. Each entity in the key-distribution process knows the identifier associated with the other entity(ies),

e. The keys are correctly associated with the entities involved in the key-distribution process, and

f. The keys have been received correctly.

8.1.5.2.3 Key Agreement

Key agreement is used in a communication environment to establish keying material using information contributed by all entities in the communication (most commonly, only two entities) without actually sending the keying material. Only **approved** key-agreement schemes **shall** be used. **Approved** key-agreement schemes using asymmetric techniques are provided in [SP800-56A] and [SP800-56B]. Key agreement uses asymmetric key pairs to calculate shared secrets, which are then used to derive symmetric keys and other keying material (e.g., IVs).

A key-agreement scheme uses either static or ephemeral asymmetric key pairs or both. The asymmetric key pairs **should** be generated and distributed as discussed in Section 8.1.5.1.

Keying material derived from a key-agreement scheme **shall** be protected as specified in Section 6.

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

1. The identifiers for entities involved in the key-establishment protocol are correctly associated with those entities. Assurance for the association of identifiers to entities may be achieved by the key-agreement scheme or may be achieved by the protocol in which key agreement is performed. Note that the identifier may be a "pseudo-identifier", not the identifier appearing on the entity's birth certificate, for example.

   In the general case, an identifier is associated with each party involved in the key-establishment protocol, and each entity in the key-establishment process must be able to associate all the other entities with their appropriate identifier. In special cases, such as the secure distribution of public information on a web site, the association with an identifier may only be required for a subset of the entities (e.g., only the server).

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 the three assurances described above have been achieved.

8.1.5.3 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 6.2.

8.1.5.3.1 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 substantial 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 vouches for the parameters (e.g., a CA), or by the entities themselves. Assurance of domain-parameter validity is addressed in [SP800-89] and [SP800-56A]. Obtaining this assurance **should** be addressed in a CA's certification practices statement or an organization's security plan.

8.1.5.3.2 Initialization Vectors

Initialization vectors (IVs) are used by symmetric-key algorithms in several modes of operation for encryption and decryption, for authentication, or both. The criteria for the generation and use of IVs are provided in the [SP800-38] series of publications; IVs **shall** be protected as specified in Section 6 of this Recommendation (i.e., SP 800-57, Part 1). 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 cryptographic mechanism.

8.1.5.3.3 Shared Secrets

Shared secrets are computed during an asymmetric key-agreement scheme and are subsequently used to derive keying material. Shared secrets are generated as specified by an appropriate key-agreement scheme (see [SP800-56A] and [SP800-56B]), and **shall not** be used directly as keying material.

8.1.5.3.4 RBG Seeds

A Random Bit Generator (RBG) is a device or algorithm that outputs a sequence of bits that is unpredictable; RBGs are often called Random Number Generators. **Approved** RBGs are specified in [SP800-90]. RBGs depend on the introduction of truly random bits called seeds, which are used to initialize an RBG and that must be kept secret. An initialized RBG is often used to generate keys and other values requiring unpredictability. The seeds themselves **shall not** be used for any purpose other than RBG input. Seeds **shall** only be transmitted using secure channels that protect the confidentiality and integrity of the seeds, as well as providing replay protection[43] and mutual authentication[44].

8.1.5.3.5 Other Public and Secret Information

Public and secret information may be used during the seeding of an RBG (see Section 8.1.5.3.4) or during the generation or establishment of keying material (see [SP800-56A], [SP800-56B] and [SP800-108]). Public information may be distributed; secret information **shall** be protected in the same manner as a private or secret key during distribution.

8.1.5.3.6 Intermediate Results

Intermediate results occur during computation using cryptographic algorithms. These results **shall not** be distributed as or with the keying material.

8.1.5.3.7 Random Bits/Numbers

Random bits (or numbers) are used for many purposes, including the generation of keys and nonces, and the issuing of challenges during communication protocols. Random bits may be distributed, but whether or not confidentiality protection is required depends on the context in which the random bits are used.

8.1.5.3.8 Passwords

Passwords are used for identity authentication or authorization, and, in some cases, to derive keying material (see [SP800-132]). Passwords may be distributed, but their protection during distribution **shall** be consistent with the protection required for their use. For example, if the password will be used to access cryptographic keys that are used to provide 128 bits of security strength when protecting data, then the password needs to be provided with at least 128 bits of protection as well. Note that poorly selected passwords may not themselves provide the required amount of protection for key access and are potentially the weak point of the process; i.e., it may be far easier to guess the password than to attempt to “break” the cryptographic protection used on the password. It is the responsibility of users and organizations to select passwords that provide the requisite amount of protection for the keys they protect.

---

[43] Assurance that a valid data transmission is not maliciously or fraudulently repeated or delayed.

[44] Authentication by each party in a transaction of the identity of the other party.

8.1.6 Key Registration Function

Key registration results in the binding of keying material to information associated with a particular entity. Keys that would be registered include the public key of an asymmetric key pair and the symmetric key used to bootstrap an entity into a system. Normally, keys generated during communications (e.g., using key-agreement schemes or key derivation functions) would not be registered. Information provided during registration typically includes the identifier of the entity associated with the keying material and the intended use of the keying material (e.g., as a signing key, data-encryption key, etc.). Additional information may include authorization information or specify a level of trust. The binding is performed after the entity's identity has been authenticated by a means that is consistent with the system policy (see Section 8.1.1). 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). Identifiers issued by a trusted third party **shall** be unique to that party.

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

When a CA performs the binding, the public key and associated information (often called *attributes*) are placed in a public-key certificate, which is digitally signed by the CA. In this case, the registered keying material may be made publicly available.

When a CA provides a certificate for a public key, the public key **shall** be verified to ensure that it is associated with the private key known by the purported owner of the public key. This provides assurance of possession. When POP is used to obtain assurance of possession, the assurance **shall** be accomplished as specified in Section 8.1.5.1.1.2.

8.2 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 6.2.2. During normal use, the keying material is stored either on the device or module that uses that material, or on an immediately 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 8.2.2.1), or the system **shall** be designed to allow reconstruction (e.g., re-derivation) of the keying material. Retrieving or reconstructing keying material from backup or an archive is commonly known as key recovery (see Section 8.2.2.2).

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 re-keying (see Section 8.2.3.1) or by key derivation (see Section 8.2.4). A key **shall** be destroyed in accordance with Section

8.3.4 and **should** be destroyed as soon as that key is no longer needed in order to reduce the risk of exposure.

8.2.1 Normal Operational Storage Function

One 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).

8.2.1.1 Cryptographic Module Storage

Keying material may be stored in the cryptographic module that adds, checks, or removes the cryptographic protection on information. The storage of the keying material **shall** be consistent with Section 6.2.2, as well as with [FIPS140].

8.2.1.2 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 6.2.2 **shall** apply to this keying material.

8.2.2 Continuity of Operations Function

Keying material can become lost or unusable, due to hardware damage, corruption or loss of program or data files, 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 backup storage. However, if operations can be continued without the backup of keying material (e.g., by re-keying), 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 cryptographically related information.

The compromise of keying material affects the continuity of operations (see Section 8.4). When keying material is compromised, the continuity of operations requires the establishment of entirely new keying material (see Section 8.1.5), following an assessment of what keying material is affected and needs to be replaced.

8.2.2.1 Backup Storage

The backup of keying material on an independent, secure storage media provides a source for key recovery (see Section 8.2.2.2). Backup storage is used to store copies of information that are also currently available in normal operational storage during a key's cryptoperiod (i.e., in the cryptographic module, or on an immediately accessible storage media - see Section 8.2.1.1). Not all keys need be backed up. The storage requirements of Section 6.2.2 apply to keying material that is backed up. Tables 7 and 8 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 the backup of 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 8.2.1). 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 in backup storage **shall** be destroyed in accordance with Section 8.3.4.

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

Table 7: Backup of keys

Type of Key	Backup?
Private signature key	No (in general); support for non-repudiation would be in question. However, backup may be warranted in some cases − a CA's private signing key, for example. When required, any backed up keys **shall** 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; if required by an application.
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; its presence in a public-key certificate that is available elsewhere may be sufficient.
Symmetric key-agreement key	OK
Private static key-agreement key	OK
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	OK
Symmetric authorization key	OK
Private authorization key	OK

Type of Key	Backup?
Public authorization key	OK; its presence in a public-key certificate that is available elsewhere may be sufficient.

Table 8: Backup of other cryptographic or related information

Type of Keying Material	Backup?
Domain parameters	OK
Initialization vector	OK, if necessary
Shared secret	No
RBG seed	No
Other public information	OK
Other secret information	OK
Intermediate results	No
Key control information (e.g., IDs, purpose, etc.)	OK
Random number	Depends on the application or use of the random number.
Passwords	OK when used to derive keys or to detect the reuse of passwords; otherwise, No
Audit information	OK

8.2.2.2 Key Recovery Function

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 or symmetric data-encryption key);
2. The application in which the key will be used (e.g., interactive communications or 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-encryption key shared by two entities);
4. The role of the entity in a communication (e.g., sender or 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)[45].

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 may be replaced in order to limit the exposure of the data protected by that key (see Section 8.2.3).

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.

8.2.3 Key Change Function

Key change is the replacement of a key with another key that performs the same function as the original key. There are several reasons for changing a 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.

8.2.3.1 Re-keying

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 re-keying. This replacement **shall** be accomplished using one of the key-establishment methods discussed in Section 8.1.5. Re-keying is used when a key has been compromised (provided that the re-keying scheme itself is not compromised) or when the cryptoperiod is nearing expiration.

8.2.3.2 Key Update Function

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). Key update is a special case of key derivation (see Section 8.2.4), where the derived key replaces the key used to derive it. For example, suppose that K_1 is used as an encryption key. When K_1 needs to be replaced, it is used to derive K_2. K_2 is then used as the new encryption key until it is replaced by K_3, which is derived from K_2.

Key update could result in a security exposure if an adversary obtains a key in the chain of keys and knows the update process used; keys subsequent to the compromised key could easily be determined.

Federal applications **shall not** use key update (also, see [SP800-152]).

8.2.4 Key Derivation Methods

Cryptographic keys may be derived from a secret value. The secret value, together with other information, is input into a key-derivation method (e.g., a key-derivation function) that outputs

[45] This could be the case when performing a key-establishment process for some key-establishment schemes (see [SP800-56A] and [SP800-56B]).

the required key(s). In contrast to key change, the derived keys are often used for new purposes, rather than for replacing the secret values from which they are derived. The derivation method **shall** be non-reversible (i.e., a one-way function) so that the secret value cannot be determined from the derived keys. In addition, it **shall not** be possible to determine a derived key from other derived keys. It should be noted that the strength of a derived key is no greater than the strength of the derivation algorithm and the secret value from which the key is derived.

Three commonly used key-derivation cases are discussed below.

1. *Two parties derive common keys from a common shared secret.* This approach is used in the key-establishment techniques specified in [SP800-56A] and [SP800-56B]. The security of this process is dependent on the security of the shared secret and the specific key-derivation method used. If the shared secret is known, the derived keys may be determined. A key-derivation method specified or allowed in [SP800-56A], [SP800-56B] or [SP800-56C] **shall** be used for this purpose. These derived keys may be used to provide the same confidentiality, identity authentication, and source authentication services as randomly generated keys, with a security strength determined by the scheme and key pairs used to generate the shared secret.

2. *Keys derived from a key-derivation key* (*master key*). This is often accomplished by using the key-derivation key, entity ID, and other known information as input to a function that generates the keys. One of the key-derivation functions defined in [SP800-108] **shall** be used for this purpose. The security of this process depends upon the security of the key-derivation key and the key-derivation function. If the key-derivation key is known by an adversary, he can generate any of the derived keys. Therefore, keys derived from a key-derivation key are only as secure as the key-derivation key itself. As long as the key-derivation key is kept secret, the derived keys may be used in the same manner as randomly generated keys.

3. *Keys derived from a password*. A user-generated password, by its very nature, is less random (i.e., has lower entropy) than is required for a cryptographic key; that is, the number of passwords that are likely to be used to derive a key is significantly smaller than the number of keys that are possible for a given key size. In order to increase the difficulty of exhaustively searching the likely passwords, a key-derivation function is iterated a large number of times. The key is derived using a password, entity ID, and other known information as input to the key-derivation function. The security of the derived key depends upon the security of the password and the key-derivation process. If the password is known or can be guessed, then the corresponding derived key can be generated. Therefore, keys derived in this manner are likely to be less secure than randomly generated keys or keys derived from a shared secret or key-derivation key. For storage applications, one of the key-derivation methods specified in [SP800-132] **shall** be used to derive keys. For non-storage applications, keys derived in this manner **shall** be used for integrity, and source authentication purposes only and not for general encryption.

8.3 Post-Operational Phase

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

8.3.1 Archive Storage and Key Recovery Functions

A key archive is a repository containing keying material and other related information for recovery beyond the cryptoperiod of the keys. 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 [SP800-57, Part 2]).

The archive **shall** continue to provide the appropriate protections for each key and any other related information in the archive, as specified in Section 6.2.2. The archive will require a strong access-control mechanism to limit access to only authorized entities. When keying material is entered into the archive, it is often time-stamped so that the date-of-entry can be determined. This date may itself be cryptographically protected so that it cannot be changed without detection.

If keying material needs to be recoverable (e.g., after the end of its cryptoperiod), either the keying material **shall** be archived, or the system **shall** be designed to allow reconstruction (e.g., re-derivation) of the keying material from archived information. Retrieving the keying material from archive storage or by reconstruction is commonly known as key recovery. The archive **shall** be maintained by a trusted party (e.g., the organization associated with the keying material or a trusted third party).

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 from time-to-time. 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 archive be backed up). For critical information that is encrypted under archived keys, it may be necessary to back up the archived keys and to store multiple copies of these archived keys in separate locations.

When archived, keying material **should** be archived prior to the end of the cryptoperiod of the key. For example, it may be prudent to archive the keying material during key activation. When no longer required, the keying material **shall** be destroyed in accordance with Section 8.3.4.

The confidentiality of archived information 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. Note that the algorithm with which the archive-encryption key is used may also provide integrity protection for the encrypted information. 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. This may impose a significant burden; therefore, the strength of the cryptographic algorithm and archive-encryption key **shall** be selected to minimize the need for re-encryption.

When the archive-encryption key and its associated algorithm do not also provide integrity protection for the encrypted information, integrity protection **shall** be provided by a separate archive-integrity key (i.e., one or more authentication or digital-signature keys that are used exclusively for the archive) or by another key that has been archived. If integrity protection is to be maintained 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.

When the confidentiality and integrity protection of the archived information is provided using separate processes, the archive-encryption key and archive-integrity key (when used) **shall** be different from each other (e.g., independently generated), and **shall** be protected in the same manner as their key type (see Section 6). Note that these two services can also be provided using authenticated encryption, which uses a single cryptographic algorithm operation and a single key.

Tables 9 and 10 indicate the appropriateness of archiving keys and other cryptographically related 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 9: 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 associated private key
Symmetric authentication key	OK	Until no longer needed to authenticate data or an identity.
Private authentication key	No	
Public authentication key	OK	
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
Public key-transport key	OK	

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

3188

3189 **Table 10: Archive of other cryptographic 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	
RBG seed	No	
Other public information	OK	Until no longer needed to process data using the public information
Other secret information	OK	Until no longer needed to process data using the secret information
Intermediate result	No	
Key control information (e.g., IDs, purpose)	OK	Until the associated key is removed from the archive
Random number		Depends on the application or use of the random number

Password	OK when used to derive keys or to detect the reuse of passwords; otherwise, No	Until no longer needed to (re-)derive keys or to detect password reuse
Audit information	OK	Until no longer needed

The 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 other archived data; recovered keys **shall not** be used to apply cryptographic protection. The key recovery process results in retrieving or reconstructing the desired keying material from archive storage in order to perform the required cryptographic operation. Immediately after completing this operation, the keying material **shall** be erased from the cryptographic process[46] for which it was recovered (i.e., it **shall not** be used for normal operational activities). However, the key **shall** be retained in the archive (see Section 8.3.4) as long as needed. Further advice on key recovery issues is provided in Appendix B.

8.3.2 Entity De-registration Function

The entity de-registration function removes the authorizations of an entity to participate in a security domain. 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 that 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, the identification information assigned to his son "John Wilson", who is hired the following Monday, **should** be different.

8.3.3 Key De-registration Function

Registered keying material may be associated with the identity of a key owner, owner information (e.g., email address), 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) by the appropriate trusted third party. In general, this will be the trusted third party that registered the key (see Section 8.1.6).

Keying material **should** be de-registered when the information associated with an entity is 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 information has become invalid. Unlike the case of a key compromise, the entity

[46] For example, an archived symmetric key could be recovered to decrypt a single message or file, or could be used to decrypt multiple messages or files, all of which were encrypted using that key during its originator-usage period.

could safely re-register the public key after modifying the entity's information through the user registration process (see Section 8.1.1).

When a registered cryptographic key is compromised, that key and any associated keying material **shall** be de-registered. When the compromised key is the private part of a public-private key pair, the public key **shall** also be revoked (see Section 8.3.5). If the registration information associated with a public-private key pair is changed, but the private key has not been compromised, the public key **should** be revoked with an appropriate reason code (see Section 8.3.5).

8.3.4 Key Destruction Function

When copies of cryptographic keys are made, care should be taken to provide for their eventual destruction. All copies of the private or symmetric key **shall** be destroyed as soon as they are no longer required (e.g., for archival or reconstruction activity) in order to minimize the risk of a compromise. Keys **shall** be destroyed in a manner that removes all traces of the keying material so that it cannot be recovered by either physical or electronic means[47]. Public keys may be retained or destroyed, as desired.

8.3.5 Key Revocation Function

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 corresponding to the 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 (excluding the key itself), the date and time of revocation and the reason for revocation, when appropriate (e.g., a key compromise). Based on the revocation information provided, other entities could then make a determination of how they will treat information protected by the revoked keying material.

For example, if a public signature-verification key is revoked because an entity left an organization, it may be appropriate to honor all signatures created prior to the revocation date (i.e., to continue to verify those signatures and accept them as valid if the verification is successful). If a signing private key is compromised, resulting in the revocation of the corresponding public key, an assessment needs to be made as to whether or not information signed prior to the revocation notice would be considered as valid.

[47] A simple deletion of the keying material might not completely obliterate the information. For example, erasing the information might require overwriting that information multiple times with other non-related information, such as random bits, or all zero or one bits. Keys stored in memory for a long time can become "burned in". This can be mitigated by splitting the key into components that are frequently updated (see [DiCrescenzo]).

As another example, a symmetric key that is used to generate MACs may be revoked so that it will not be 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 using the same encryption key), the entity revoking the key **shall** inform the other entity of the revocation. 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 needs to be revoked (e.g., using a certificate revocation request). The infrastructure **shall** respond by de-registering the key material (see Section 8.3.3).

In a PKI, key revocation is commonly achieved by including the certificate in a list of revoked certificates (i.e., a CRL). If the PKI uses online status mechanisms (e.g., the Online Certificate Status Protocol [RFC 2560]), revocation is achieved by informing the appropriate certificate status server(s). For example, when a private key is compromised, the corresponding public-key certificate **shall** be revoked as soon as possible. Certificate revocation because of a key compromise indicates that the binding between the owner and the key is no longer to be trusted; relying parties **should not** accept the certificate without seriously considering the risks and consulting the organization's policy about this situation. Other revocation reasons indicate that, even though the original binding may still be valid and the key was not compromised, the use of the public key in the certificate **should** be terminated; again, the relying party **should** consult his organization's policy on this issue.

In a symmetric-key system, key revocation could, in theory, be achieved by simply deleting the key from the server's storage. Key revocation for symmetric keys is more commonly achieved by adding the key to a blacklist or compromised key list; this helps satisfy auditing and management requirements.

8.4 Destroyed Phase

The keying material is no longer available. All records of its existence may have been deleted, though this is not required. Some organizations may require the retention of certain key metadata elements for audit purposes. For example, if a copy of an ostensibly destroyed key is found in an uncontrolled environment or is later determined to have been compromised, records of the identifier of the key, its type, and its cryptoperiod may be helpful in determining what information was protected under the key and how best to recover from the compromise.

In addition, by keeping a record of the metadata of both destroyed and compromised keys, one will be able to track which keys transitioned through a normal lifecycle and which ones were compromised at some time during their lifecycle. Thus, protected information that is linked to key names that went through the normal lifecycle may still be considered secure, provided that the security strength of the algorithm remains sufficient. However, any protected information that is linked to a key name that has been compromised may itself be compromised.

9 Accountability, Audit, and Survivability

Systems that process valuable information require controls in order to protect the information from unauthorized disclosure and modification. Cryptographic systems that contain keys and other cryptographic information are especially critical. Three useful control principles and their application to the protection of keying material are highlighted in this section.

9.1 Accountability

Accountability involves the identification of those entities that 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 when they are detected. Although it is preferred that no humans be able to view keys, 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 to access or control 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 lifespan. This would include the person in charge of generating the key, the person who used the key to cryptographically protect data, anyone else known to have accessed the key, and the person who was responsible for destroying the key when it was no longer needed. Even though these individuals may never have actually seen 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 a 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 be applicable to all systems or all types of keys. Some of the principles apply to long-term keys that are controlled by humans. The principles include:

a. Uniquely identifying keys;

b. Identifying the key user;

c. Identifying the 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.

9.2 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 [SP800-57, 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. Highly unusual events **should** be noted and reviewed as possible indicators of attempted attacks on the system.

9.3 Key Management System Survivability

9.3.1 Backup Keys

[OMB11/01] 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 backed up or archived copies of the keys necessary to decrypt stored enciphered information, including master keys, key-wrapping 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 7 and 8 in Section 8.2.2.1).

As the tables in Section 8.2.2.1 show, there are other operational keys in addition to those associated with decryption that organizations may need to backup (e.g. public signature-verification keys and authorization keys). Backed up or archived copies of keying material **shall** be stored in accordance with the provisions of Section 6 in order to protect the confidentiality of encrypted information and the integrity of source authentication, integrity authentication, and authorization processes.

9.3.2 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. When will archiving be done (e.g., during key activation or at the end of a key's cryptoperiod)?
4. Who will be responsible for protecting the backed-up or archived keying material?
5. What procedures need to be put in place for storing and recovering the keying material?
6. Who can request a recovery of the keying material and under what conditions?
7. Who will be notified when a key recovery has taken place and under what conditions?
8. What audit or accounting functions need to be performed to ensure that the keying material is only provided to authorized entities?

9.3.3 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 encrypted by a strong algorithm may not be practical without the correct decryption key. The 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.

9.3.3.1 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 done. 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 (CRLs) 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 for 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;
11. Loss or corruption of the association of keying material to the owners/users of the keying material; and
12. Unavailability of older software or hardware that is needed to access keying material or process protected information.

While recovery discussions most commonly focus on the recovery of encrypted data and the restoration of encrypted communication capabilities, planning **should** also address 1) the restoration of access (without creating a temporary loss of access protections) where cryptography is used in access control mechanisms, 2) the restoration of critical processes (without creating a 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 the 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.

9.3.3.2 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 the Office of Management and Budget has noted in [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. More detail regarding contingency plans and planning requirements is provided in Part 2 of this *Recommendation for Key Management* [SP800-57, Part 2].

9.3.4 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 compromised keys **shall** be revoked; 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 **should** be undertaken.

Where symmetric keys or private (asymmetric) keys are used to protect only a single user's local information or communications between a single pair of users, the compromise recovery process can be relatively simple and inexpensive. 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. A CA's private signature key, especially if it is used to sign a root certificate in a public-key infrastructure;
2. A symmetric key-wrapping key shared by a large number of users;
3. A private asymmetric key-transport key shared by a large number of users;
4. A master key used in the derivation of keys by a large number of users;
5. A symmetric data-encryption key used to encrypt data in a large distributed database;
6. A symmetric key shared by a large number of communications network participants; and
7. A key used to protect a large number of stored keys.

In all of these cases, a large number of key owners or relying parties (e.g., all parties authorized to use the secret key of a symmetric-key algorithm or the public key of an asymmetric-key algorithm) would need to be immediately notified of the compromise. The inclusion of the key identifier on a Compromised Key List (CKL) or the certificate serial number on a Certificate Revocation List (CRL) to be published at a later date might not be sufficient. This means that a list of (the most-likely) affected entities might need to be maintained, and a means for communicating news of the compromise would be required. Particularly in the case of the compromise of a symmetric key, news of the compromise and the replacement of keys **should** be sent only to the affected entities so as not to encourage others to exploit the situation.

In all of these cases, a secure path for replacing the compromised keys is required. In order to permit rapid restoration of service, an automated (e.g., over-the-air or network-based) replacement path is preferred (see Section 8.2.3). In some cases, however, there may be no practical alternative to manual distribution (e.g., the compromise of a root CA's private key). A contingency distribution of alternate keys may help restore service rapidly in some circumstances (e.g., the compromise of a widely held symmetric key), but the possibility of a simultaneous compromise of operational and contingency keys would need to be considered.

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

10 Key Management Specifications for Cryptographic Devices or Applications

Key management is often an afterthought in the cryptographic development process. As a result, cryptographic subsystems often fail to support the key management functionality and protocols that are necessary to provide adequate security with the minimum necessary reduction in operational efficiency. All cryptographic development activities **should** involve key management planning and specification (see [SP800-57, Part 2]) by those managers responsible for the secure implementation of cryptography into an information system. Key management planning **should** begin during the initial conceptual/development stages of the cryptographic development lifecycle, or during the initial discussion stages for the application 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 (see [SP800-130] and [SP800152]).

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 mechanisms 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 recommendations. 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 recommendations.

10.1 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 *Recommendation for Key Management* [SP800-57, Part 2]).

10.2 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 identifier, and the developer's or integrator's identifier. A revision page, a list of reference documents, a table of contents, and a definition of abbreviations and acronyms page **should** also be included. The terminology used in a Key Management Specification **shall** be in accordance with the terms defined in appropriate NIST standards and guidelines. Unless the information is tightly controlled, the Key Management Specification **should not** contain proprietary or sensitive 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 sections below.]

10.2.1 Cryptographic Application

A Cryptographic Application section provides a basis for the development of the rest of the Key Management Specification. The Cryptographic Application 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 authentication, source authentication, non-repudiation support, access control, 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.

10.2.2 Communications Environment

A 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 (e.g., intranet, Internet, VPN);
2. Wired communications (e.g., landline, dedicated or shared switching resources); and
3. Wireless communications (e.g., cell phones).

The environment may also include any anticipated access controls on communications resources, data sensitivity, privacy issues, etc.

10.2.3 Key Management Component Requirements

A Key Management Component Requirements section describes the types and logical structure of the keying material required for the operation of the cryptographic device. Cryptographic applications using public-key certificates (e.g., 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 (e.g., for PKI: CA, signature, key establishment, and authentication);
2. The key management algorithm(s) (the applicable **approved** algorithms);
3. The keying material format(s) (reference any existing key specification, if known);
4. The set of acceptable PKI policies (as applicable); and
5. The tokens to be used.

The description of the key-management component format may reference a key specification for an existing cryptographic device. 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.

10.2.4 Key Management Component Generation

The Key Management Specification **should** include a description of the 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, the key-management components that will be required from external sources **should** be identified.

10.2.5 Key Management Component Distribution

When a device supports the automated distribution of keying material, the Key Management Specification **should** include a description of the distribution method(s) (where employed) used for keying material supported by the device. The distribution plan may describe the circumstances under which the key-management components are encrypted or in plaintext, their physical form (electronic, 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.

10.2.6 Keying Material Storage

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.

10.2.7 Access Control

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 and identity-based privileging, and the use of any tokens **should** be described.

10.2.8 Accounting

The Key Management Specification **should** describe any device or application support for the accounting of the 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 multi-party control is required, if applicable. Section 9.1 of this Recommendation provides accountability guidance.

10.2.9 Compromise Management and Recovery

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 for re-keying. 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 9.3.4 of this Recommendation.

10.2.10 Key Recovery

The Key Management Specification **should** include a description of 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 to 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 described. General contingency planning guidance is provided in Section 9.3.3 of this Recommendation. Key recovery is treated in detail in Appendix B.

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

Integrity and source authentication services are particularly important in protocols that include key management. When integrity or source authentication services are discussed in this Recommendation, they are afforded by "strong" cryptographic integrity or source authentication mechanisms. Secure communications and key management are typically provided using a communication protocol that offers certain services, such as integrity protection or a "reliable" transport service[48]. However, the integrity protection or reliable transport services of communication 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 communication channels have some noise (i.e., unintentional errors inserted by the transmission media), and other factors, such as network congestion, can cause network packets[49] to be lost. Therefore, integrity protection and reliable transport services for communication 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[50] 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 that contain bit errors) or lost packets may request the sender to retransmit them. The non-cryptographic checksums are generally effective at detecting transmission noise. For example, the common CRC-32 checksum algorithm used in local-area network 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 as checksums, depending on the application and noise properties of the channel.

Cryptographic integrity authentication mechanisms (e.g., MACs or digital signatures), on the other hand, protect against an active, intelligent attacker who might attempt to disguise his attack as noise. Typically, the bits altered by the attacker are not random; they are targeted at

[48] A means of transmitting information within a network using protocols that provide assurances that the information is received correctly.

[49] A formatted unit of data used to send messages across a network. Messages may be divided into multiple packets for transmission efficiency.

[50] 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 re-computes 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).

system properties and vulnerabilities. Cryptographic integrity authentication mechanisms are effective in detecting random noise events, but they also detect the more systematic deliberate attacks. Cryptographic hash functions, such as SHA-256 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 necessary to perform 2^{128} SHA-256 hash operations to find two messages that hash to the same value, and it is much harder to find another message whose SHA-256 hash is the same value as the hash of any given message. Cryptographic message authentication code (MAC) algorithms employ hash functions or symmetric encryption algorithms and keys to authenticate the source of a message and to protect the integrity of a message (i.e., to detect errors). Digital signatures use public-key algorithms and hash functions to provide both integrity and source authentication services. Compared to non-cryptographic integrity or source authentication mechanisms, these cryptographic services are usually computationally more expensive; this seems to be unavoidable, since cryptographic protections must also resist deliberate attacks by knowledgeable adversaries with substantial resources.

Cryptographic and non-cryptographic integrity authentication mechanisms may be used together. For example, consider the TLS protocol (see [SP800-52]). In TLS, a client and a server can authenticate the identity of each other, establish a shared "master key" and transfer encrypted payload data. Every step in the entire TLS protocol run is protected by cryptographic integrity and source 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 with a given probability. 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 authentication mechanisms, but a sophisticated attacker could easily fool the weaker non-cryptographic checksums of TCP. However, because of the cryptographic integrity authentication mechanism provided in TLS, the attack is thwarted.

There are some interactions between cryptographic 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[51]

[51] Stream ciphers encrypt and decrypt one element (e.g., bit or byte) at a time. There are no **approved** 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.

with non-cryptographic checksums (e.g. CRC-32) that are computed over the plaintext data 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 certain bits of the underlying plaintext. 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, integrity authentication, source authentication, non-repudiation support or access control. Policies **shall** be established to address the protection and continued accessibility of cryptographically protected information, and procedures **shall** be in place to ensure that the information remains viable during its lifetime. When cryptographic keying material is 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 MAC) 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 plaintext information.

In order to have this keying material available when required, the keying material needs to be saved somewhere or to be constructible (e.g., 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 8.2.1), e.g., the verification of MACs for archived documents. 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.

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 [SP800-57, Part 2]). If the decision is made to provide key recovery, the appropriate method of key recovery **should** be selected, designed and implemented, based on the type of keying material to be recovered; an appropriate entity needs to be selected to maintain the backup or archive database and manage the key recovery process.

If the decision is made to provide key recovery for a key, all information associated with that key **shall** also be recoverable (see Table 5 in Section 6).

B.1 Recovery from Stored Keying Material

The primary purpose of the back up or archiving of keying material is to be able to recover that material when it is not otherwise available. For example, encrypted information cannot be transformed back into plaintext information if the decryption key is lost or modified; the integrity of data cannot be authenticated if the key used to verify the integrity of that data is not available. The key recovery process retrieves the keying material from backup or archive storage, and places it either in a device or module, or in other immediately accessible storage (see Section 8.3.1).

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 normal operational storage (see Section 8.2.1), backup storage (see Section 8.2.2.1) or archive storage (see Section 8.3.1).

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

The decision as to whether to back up or archive keying material for possible key recovery **should** be made on a case-by-case basis. The decision **should** be based on the list provided in Section 8.2.2.2.

When the key-recovery operation is requested by the key's owner, the following actions **shall** be taken:

1. If the key is lost with the possibility of having been compromised, then the key **shall** be replaced as soon as possible after recovery in order to limit the exposure of the recovered key and the data it protects (see Section 8.2.3.1). This requires reapplying the protection on the protected data using the new key. For example, suppose that the key that was used to encrypt data (Key_A) has been misplaced in a manner in which it could have been compromised. As soon as possible after Key_A is recovered, Key_A **shall** be used to decrypt the data, and the data **shall** be re-encrypted under a new key (Key_B). Key_B **shall** have no relationship to Key_A (e.g., Key_B **shall not** be an update of Key_A).

2. If the key becomes inaccessible or has been modified, but compromise is not suspected, then the key may be recovered. No further action is required (e.g., re-encrypting the data). For example, if the key becomes inaccessible because the system containing the key crashes, or the key is inadvertently overwritten, and a compromise is not suspected, then the key may simply be restored.

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 (e.g., the metadata associated with the key) **shall** 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 corresponding public key (the public signature-verification key) is used by relying entities to verify the digital signature.

B.3.1.1 Private Signature Keys

Private signature keys **shall not** be archived (see Table 9 in Section 8.3.1). Key backup is not usually desirable for the private key of a signing key pair, since support for the non-repudiability of the signature comes into question. However, exceptions may exist. For example, replacing the private signature key and having its corresponding public signature-verification key distributed (in accordance with Section 8.1.5.1) in a timely manner may not be possible under some circumstances, so recovering the private signature key from backup storage may be justified. This may be the case, for example, for the private signature key of a CA.

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 new key pair, and certify and distribute a new public signature-verification key. 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 corresponding public key could be generated, and the public key distributed in accordance with Section 8.1.5.1 for use if the primary private signature key becomes unavailable.

B.3.1.2 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 corresponding private signature key. In the case of a public key that has been certified (e.g., by a Certification 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 private key's cryptoperiod, and **should** be stored in archive storage as long as required for the verification of signed data.

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 integrity (and possibly the source) 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 may not be performed prior to storage.

For each of the above cases, a decision to provide a key recovery capability **should** be made, based on the following considerations.

In case 1, the symmetric authentication key need not be backed up or archived 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 and source of the information needs to be determined.

In case 3, the symmetric authentication key need not be backed up or archived 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 base 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 originating entity. The corresponding 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 support for 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 identity of the entity that is participating in an authenticated communication session.

In the case of a public key that has been certified (e.g., by a Certification 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

The private key is used to establish the identity of an entity who is participating in an authenticated communication session. The private authentication key need not be backed up if a new key pair can be generated and distributed in accordance with Section 8.1.5.1 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.

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. The same key is used initially to encrypt the plaintext information to be protected, and later to decrypt the encrypted information (i.e., the ciphertext), thus obtaining the original plaintext.

The key needs to be available for as long as any information that is encrypted using that key may need to be decrypted. 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 **should** be stored in archive storage, if required. In many cases, the key is protected and stored with the encrypted data. When archived, the key is wrapped (i.e., encrypted) by an archive-encryption key or by a symmetric key-wrapping key that is wrapped by a protected archive-encryption key.

A symmetric-data encryption key that is used only for transmission 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 may not need to 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 transmitted 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 may not be necessary.

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

B.3.6 Random Number Generation Keys

A key used for deterministic random bit generation **shall 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 automated key-establishment protocols, then the backup or archiving of the master key may not be necessary or 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 a 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. The transported keying material could be stored in its encrypted form for decryption at a later time. The originating entity in a communication 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 used during communications without storing the encrypted keying material, 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. Alternatively, one or more additional key pairs could be made available (i.e., already generated and distributed). Otherwise, the private key **should** be backed up. The private key-transport key may be archived.

If the transported keying material is stored in its encrypted form, 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

Backup or archiving of the public key may be done, but may not be necessary.

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 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 automated 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 may be archived.

B.3.10 Static Key-Agreement Key Pairs

Static key-agreement key pairs are used to establish shared secrets between entities (see [SP800-56A] and [SP800-56B]), sometimes in conjunction with ephemeral key pairs (see [SP800-56A]). Each entity uses its private key-agreement key(s), the other entity's public key-agreement key(s) and possibly its 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 [SP800-56A] and [SP800-56B]).

B.3.10.1 Private Static Key-Agreement Keys

If the private static key-agreement key cannot be replaced in a timely manner, or if it needs to be retained in order to recover encrypted stored data, then the private key **should** be backed up in order to continue operations. The private key may be archived.

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 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 may be archived.

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 [SP800-56A]).

B.3.11.1 Private Ephemeral Keys

Private ephemeral keys **shall not**[52] 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. This may allow the reconstruction of the established keying material, as long as 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). The loss of these keys will deny the privileges (e.g., prohibit access and disallow the 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 symmetric authorization key **shall not** be archived.

B.3.13 Authorization Key Pairs

Authorization key pairs are used to determine the privileges that an entity may assume. 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.

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

B.3.13.1 Private Authorization Keys

The loss of the private authorization key will deny privileges (e.g., prohibit access and disallow the performance of certain functions requiring authorization). 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.

B.3.13.2 Public Authorization Keys

If the authorization key pair can be replaced in a timely fashion (i.e., by a 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. There is no restriction about archiving the public key.

B.3.14 Other Cryptographically Related Material

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

B.3.14.1 Domain Parameters

Domain parameters are used in conjunction with some public key algorithms to generate key pairs. They are also used with key pairs to create and verify digital signatures or to establish keying material. 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, 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.

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. IVs are often stored with the data that they protect. If not stored with the data, 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 RBG Seeds

RBG seeds are used in the generation of deterministic random bits that need to remain secret. These seeds **shall not** be shared with other entities. RBG seeds **shall not** be backed up or archived.

B.3.14.5 Other Public and Secret Information

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

B.3.14.6 Intermediate Results

The intermediate results of a cryptographic operation **shall not** be backed up or archived.

B.3.14.7 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 to identify the entities that share the key (see Section 6.2.3). This information is contained in the key's metadata (see Section 6.2.3.1).

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

B.3.14.8 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.3.14.9 Passwords

A password is used to acquire access to privileges by an entity, to derive keys or to detect the re-use of passwords.

If the password is only used to acquire access to privileges, and can be replaced in a timely fashion, then the password need not be backed up. In this case, a password **shall not** be archived.

If the password is used to derive cryptographic keys or to prevent the re-use of passwords, the password **should** be backed up and archived.

B.3.14.10 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 (e.g., the key's metadata). 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 below 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 identifier associated with the owner of the key (i.e., the individual, application or organization that created the key or that owns 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 **shall** 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 used by the Federal government **shall**:

1. Generate or provide sufficient KRI to allow recovery or verification of protected information when such information has been stored;
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. Use cryptographic modules that are compliant with [FIPS140];
5. Use **approved** algorithms, when cryptography is used;
6. Use algorithms and key lengths that provide security strengths commensurate with the sensitivity of the information associated with the KRI;
7. Be designed to enforce the Key Recovery Policy (see Appendix B.5);
8. 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. Protect the KRI from modification;
10. 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 **should** include the identification of the event being audited, the time of the event, the

identifier associated with the user causing the event, and the success or failure of the event;

11. Limit access to the KRI, the audit trail and authentication data to authorized individuals; and
12. Prohibit modification of the audit trail.

B.5 Key Recovery Policy

For each system, application and cryptographic technique used, consideration **shall** be given as to whether or not the keying material may need to be saved for later recovery to allow subsequent decryption or checking the information protected by the keying material. An organization that determines that key recovery is required for some or all of its keying material **should** develop a Key Recovery Policy that addresses the protection and continued accessibility of that information[53] (see [DOD-KRP]). The policy **should** answer the following questions (at a minimum):

1. What keying material needs to be saved for a given application? For example, keys and IVs used for the decryption of stored information may need to be saved. Keys for the authentication of stored or transmitted information may also need to be saved.
2. How and where will the keying material 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? For example, each individual, organization or sub-organization could be responsible for their own keying material, or an external organization could perform this function.
4. Who is authorized to receive the KRI upon request 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 can the policy be modified and by whom?
6. What audit capabilities and procedures will be included in the KRS? The policy **shall** 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 will the KRS deal with aged keying material whose security strength is now reduced beyond an acceptable level?
8. Who will 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

[53] An organization's key recovery policy may be included in its PKI Certificate Policy.

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. What procedures need to be followed when the KRS or some portion of the data within the KRS is compromised?

APPENDIX C: References

[AC] Applied Cryptography, Schneier, John Wiley & Sons, 1996.

[ANSX9.31] Digital Signatures using reversible Public Key Cryptography for the Financial Services Industry (rDSA), 1998 (Withdrawn).

[ANSX9.44] Public Key Cryptography for the Financial Services Industry: Key Agreement Using Factoring-Based Cryptography, August 24, 2007.

[ANSX9.62] Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), January 22, 2009.

[DiCrescenzo] How to forget a secret, G. Di Crescenzo, N. Ferguson, R. Impagliazzo, and M Jakobsson, STACS '99, Available via http://www.macfergus.com/pub/forget.html.

[DOD-KRP] Key Recovery Policy for the United States Department of Defense, Version 3.0, 31 August 2003, DoD KRP, Attn: I5P, 9800 Savage Road, STE 6737, Ft Meade, MD, 20755-6737.

[FIPS140] Federal Information Processing Standard 140-2, Security Requirements for Cryptographic Modules, May 25, 2001.

[FIPS180] Federal Information Processing Standard 180-4, Secure Hash Standard (SHS), August 2015.

[FIPS186] Federal Information Processing Standard 186-4, Digital Signature Standard (DSS), (Revision of FIPS 186-2, June 2000), July 2013.

[FIPS197] Federal Information Processing Standard 197, Advanced Encryption Standard (AES), November 2001.

[FIPS198] Federal Information Processing Standard 198-1, Keyed-Hash Message Authentication Code (HMAC), July 2008.

[FIPS199] Federal Information Processing Standard 199, Standards for Security Categorization of Federal Information and Information Systems, v 1.0, February 2004.

[FIPS202] Federal Information Processing Standard 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August 2015.

[HAC] Handbook of Applied Cryptography, Menezes, van Oorschot and Vanstone, CRC Press, 1996.

[ITLBulletin] Techniques for System and Data Recovery, NIST ITL Computer Security Bulletin, April 2002.

[OMB11/01] OMB Guidance to Federal Agencies on Data Availability and Encryption, Office of Management and Budget, November 26, 2001.

[PKCS#1] PKCS #1 v2.1, RSA Cryptography Standard, RSA Laboratories, June 14, 2002.

[RFC2560] Request for Comment 2560, X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP, IETF Standards Track, June 1999.

[SP800-14] Special Publication 800-14, Generally Accepted Principles and Practices for Securing Information Technology Systems, September 1996.

[SP800-21] Special Publication 800-21, Guideline for Implementing Cryptography in the Federal Government, December 2005.

[SP800-32] Special Publication 800-32, Introduction to Public Key Technology and the Federal PKI Infrastructure, February 2001.

[SP800-37] Special Publication 800-37, Guide for the Security Certification and Accreditation of Federal Information Systems, February 2010.

[SP800-38] Special Publication 800-38, Recommendation for Block Cipher Modes of Operation:

SP 800-38A, Methods and Techniques, December 2001.

SP 800-38A (Addendum): Three Variants of Ciphertext Stealing for CBC Mode, October 2010.

SP 800-38B: The CMAC Authentication Mode, May 2005.

SP 800-38C: The CCM Mode for Authentication and Confidentiality, May 2004.

SP 800-38D: Galois/Counter Mode (GCM) and GMAC, November 2007.

SP 800-38E: The XTS-AES Mode for Confidentiality on Storage Devices, January 2010.

SP 800-38F: Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, December 2012.

SP 800-38G: Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption, July 2013 (Draft).

[SP800-38A] Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation-Methods and Techniques, December 2001.

[SP800-38B] Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Authentication Mode, May 2005.

[SP800-38F] Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping, December 2012.

[SP800-52] Special Publication 800-52, Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations, April, 2014.

[SP800-56A] Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, May 2013.

[SP800-56B] Special Publication 800-56B, Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, September 2014.

[SP800-56C] Special Publication 800-56C, Recommendation for Key Derivation through Extraction-then-Expansion, November 2011.

[SP800-57, Part 2]

Special Publication 800-57, Part 2, Recommendation for Key Management: Part 2: Best Practices for Key Management Organization, August 2005.

[SP800-67] Special Publication 800-67, Recommendation for Triple Data Encryption Algorithm Block Cipher, January 2012.

[SP800-89] Special Publication 800-89, Recommendation for Obtaining Assurances for Digital Signature Applications, November 2006.

[SP800-90] Special Publication 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, November 2014 (Draft).

Special Publication 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation, September 2013 (Draft).

Special Publication 800-90 C: Recommendation for Random Bit Generator (RBG) Constructions, September 2013 (Draft).

[SP800-90A] Special Publication 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, June 2015.

[SP800-90B] Special Publication 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generation, September 2013 (Draft).

[SP800-90C] Special Publication 800-90C, Recommendation for Random Bit Generator (RBG) Constructions, September 2013 (Draft).

[SP800-107] Special Publication 800-107, Recommendation for Applications Using Approved Hash Algorithms, August 2012.

[SP800-108] Special Publication 800-108, Recommendation for Key Derivation Using Pseudorandom Functions, October 2009.

[SP800-131A] Special Publication 800-131A, Recommendation for the Transitioning of Cryptographic Algorithms and Key Sizes, July 2015 (Draft).

[SP800-130] Special Publication 800-130, A Framework for Designing Cryptographic Key Management Systems, August 2013.

[SP800-132] Special Publication 800-132, Recommendation for Password-Based Key Derivation - Part 1: Storage Applications, December 2010.

[SP800-133] Special Publication 800-133, Recommendation for Cryptographic Key Generation, December 2012.

[SP800-152] Special Publication 800-152, DRAFT A Profile for U. S. Federal Cryptographic Key Management Systems (CKMS), December 2014 (Draft).

APPENDIX D: Revisions

The original version of this document was published in August 2005. In May 2006, the following revisions were incorporated:

1. The definition of security strength has been revised to remove "or security level" from the first column, since this term is not used in the document.
2. In the footnote for 2TDEA in Table 2 of Section 5.6.1, the word "guarantee" has been changed to "assessment".
3. In the paragraph under Table 2 in Section 5.6.1: The change originally identified for the 2006 revision has been superseded by the 2011 revision discussed below.
4. In Table 3 of Section 5.6.1, a list of appropriate hash functions have been inserted into the HMAC and Key Derivation Function columns. In addition, a footnote has been included for the Key Derivation Function column.
5. The original text for the paragraph immediately below Table 3 has been removed.

In March 2007, the following revisions were made to allow the dual use of keys during certificate requests:

1. In Section 5.2, the following text was added:

   > "This Recommendation also permits the use of a private key-transport or key-agreement private key to generate a digital signature for the following special case:
   >
   > > When requesting the (initial) certificate for a static key-establishment key, the associated private key may be used to sign the certificate request. Also refer to Section 8.1.5.1.1.2."

2. In Section 8.1.5.1.1.2, the fourth paragraph was originally as follows:

   > "The owner provides POP 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, the owner may 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 and then provide evidence that the key was correctly decrypted (e.g., by encrypting a random challenge from the CA, then the owner has established POP. Where a key pair is intended to support key establishment, POP **shall not** be afforded by generating and verifying a digital signature with the key pair."

   The paragraph was changed to the following, where the changed text is indicated in italics:

   > "The *(reputed)* owner ***should*** *provide* POP by performing operations with the private key that satisfy the indicated key use. For example, if a key pair is intended to support *RSA* key transport, the *CA may provide the owner with a key* that is encrypted using the owner's public key. If the owner can correctly decrypt the ciphertext key using the associated private key and then provide evidence that the key was correctly decrypted (e.g., by encrypting a random

challenge from the CA, then the owner has established POP. *However, when a key pair is intended to support key establishment, POP may also be afforded by using the private key to digitally sign the certificate request (although this is not the preferred method). The private key establishment private key (i.e., the private key-agreement or key-transport key)* ***shall not*** *be used to perform signature operations after certificate issuance.*"

In September 2011, several editorial corrections and clarifications were made, and the following revisions were also made:

1. The Authority section has been updated.

2. Section 1.2: The description of SP800-57, Part 3 has been modified per that document.

3. Section 2.1: Definitions for key-derivation function, key-derivation key, key length, key size, random bit generator and user were added. Definitions for archive, key management archive, key recovery, label, owner, private key, proof of possession, public key, security life of data, seed, shared secret and **should** have been modified. The definition for cryptomodule was removed.

4. Section 2.2: The RBG acronym was inserted.

5. References to FIPS 180-3, FIPS 186-3, SP 800-38, SP 800-56A, SP 800-56B, SP 800-56C, SP 800-89, SP 800-90, SP 800-107, SP 800-108, SP 800-131A, SP 800-132 and SP 800-133 have been corrected or inserted.

6. Section 4.2.4: A footnote was added about the two general types of digital signatures and the focus for this Recommendation.

7. Sections 4.2.5, 4.2.5.3, 4.2.5.5 and 5.3: Discussions about SP 800-56B have been included.

8. Section 5.1.1: The definitions of private signature key, public signature-verification key, symmetric authentication key, private authentication key and public authentication key have been corrected to reflect their current use in systems and protocols. This change is reflected throughout the document.

9. Section 5.1.2, item 3: The description of shared secret has been modified to state that shared secrets are to be protected and handled as if they are cryptographic keys.

10. Sections 5.1.2, 5.3.7, 6.1.2 (Table 5), 8.1.5.3.4, 8.1.5.3.5, 8.2.2.1 (Table 7) and 8.3.1 (Table 9): "Other secret information" has been added to the list of other cryptographic or related information.

11. Section 5.3.1: An additional risk factor was inserted about personnel turnover.

12. Section 5.3.4: A statement was inserted to clarify the difference between the cryptoperiod of a public key and the validity period of a certificate.

13. Section 5.3.6: Statements were inserted that emphasize that longer or shorter cryptoperiods than those suggested may be warranted. Also, further discussion was added about the cryptoperiod of the public ephemeral key-agreement key.

14. Section 5.4.4: A discussion of an owner's assurance of private-key possession was added.

15. Section 5.5: Statements were added about the compromise of a CA's private signature key, and advice was provided for handling such an event.

16. Section 5.6.1: Table 3 and the text preceding the table have been revised for clarity. Additional footnotes were inserted related to table entries, and the footnote about the security strength provided by SHA-1 was modified to indicate that its security strength for digital signature applications remains the subject of speculation.

17. Sections 5.6.2 – 5.6.4: Table 4 and the text preceding it have been modified to be consistent with SP 800-131A. Also, the examples have been modified.

18. Section 5.6.5: This new section was added to address the implications associated with the reduction of security strength because of improvements in computational capabilities or cryptanalysis.

19. Sections 7, 7.1, 7.2 and 7.3: The description of the states and their transitions have been reworded to require specific behavior (e.g., using **shall** or **shall not** statements, rather than containing statement of fact (e.g., using "is" or are").

20. Section 7.3: A discussion of the transition of a private key-transport key and an ephemeral private key-agreement key were added. The previous discussion on private and public key-agreement keys was changed to discuss static private and public key-agreement keys and ephemeral public key-agreement keys.

21. Section 8.1.5.3.4: This section was revised to be more consistent with SP 800-90A.

22. Sections 8.1.5.3.7 and 8.1.5.3.8: New sections were inserted to discuss the distribution of random numbers and passwords.

23. Section 8.1.6: Text was inserted to indicate which keys would or would not be registered.

24. Section 8.2.4: This section was revised to be consistent with SP 800-56A SP 800-56B, SP 800-56C, SP 800-108 and SP 800-132.

25. Section 8.3.1, Table 9: The table was modified to indicate that it is OK to archive the static key-agreement key.

26. Changes were made to Sections 8.3.1; 9.3.2; and Appendices B, B.1, B.3, B.3.1.2, B.3.2, B.3.4, B.3.5, and B.3.10.2 to remove the impression that archiving is only performed after the end of the cryptoperiod of a key (e.g., keys could be archived immediately upon activation), and that the keys in an archive are only of historical interest (e.g., they may be needed to decrypt data long after the cryptoperiod of a key).

27. Section 8.3.3: The discussion about de-registering compromised and non-compromised keys was modified.

28. Section 8.3.5: A discussion about how revocation is achieved for a PKI and for symmetric-key systems was added.

29. Appendix B.14.9 was revised to be consistent with SP 800-132.

30. The tags for references to FIPS were modified to remove the version number. The version number is provided in Appendix C.

In 2015, several editorial corrections and clarifications were made, and the following revisions were also made:

1. Changed the reference to SP 800-21 to SP 800-175.
2. Corrected web site links.
3. Section 1.4: Now refer to FIPS and NIST Recommendations as "NIST standards." Explain the concept of the cryptographic toolkit (in a footnote).
4. Section 2.1: Modified the definitions of Algorithm originator-usage period, Archive, authentication, authentication code, certification authority, DRBG, Digital signature, Key derivation, Key-encrypting key, Key Management Policy, Key transport, Key update, Key wrapping, Key-wrapping key, Message authentication code, Non-repudiation, Owner, Recipient-usage period, RBG seed, Secure communication protocol, Security services, Signature generation, Signature verification, Source authentication, and Trust anchor.

   Added definitions for Data-encryption key, Identity authentication, Integrity authentication, Integrity protection, Key-derivation method, Key length, NIST standards, and Source authentication.

   Removed the definitions of Key attribute and Work.
5. Section 2.2: Referenced the applicable publications.
6. Many of the mentions of "attributes" have been changed to "metadata" to align with discussions in SP 800-152.

7. Section 3 and throughout the document: more clearly discusses authentication as either integrity authentication or source authentication. Identity authentication has been considered as source authentication.
8. Section 3.3: Rewritten to more clearly discuss integrity authentication or source authentication.
9. Section 3.4: Rewritten to more clearly discuss the how authorization is obtained.
10. Section 3.5: Rewritten to provide a more realistic discussion of non-repudiation. Most references to non-repudiation in the document have been removed.
11. Inserted references to FIPS 202, as well as to FIPS 180.
12. Section 4.1: Remove a reference to the Dual_EC_DRBG specified in SP 800-90A.
13. Section 4.2.2.2: Rewritten to address the non-approval of two-key TDEA for applying protection after 2015 (as indicated in SP 800-131A).
14. Section 4.2.2.3: Inserted rationale for not using the ECB mode.
15. Section 4.2.4:Rewritten to provide more information about FIPS 186.
16. Section 4.2.5.1: Further discussion of SP 800-56A has been included.
17. Section 4.2.5.3: Added references to SP 800-56A and SP 800-56B for discussion of the security properties of the key-establishment schemes.
18. Section 4.2.5.4: Rewritten to clarify the use of "key wrapping"vs. "key encryption" in the document.
19. Section 4.2.7: Rewritten to describe SP 800-90A, SP 800-90B and SP 800-90C.
20. Section 5.1.1: More details added to the symmetric data-encryption key, symmetric key-wrapping key, and public key-transport key.

    Added notes of intent to the private and public authentication keys.
21. Section 5.2: The use of "should" in the first line has been changed to "shall" to more strongly indicate that keys must not be used for multiple purposes. The use of "should" presented a conflict with later discussions in the document.
22. Section 5.3.1: Added a reference to quantum computers in the list.
23. Section 5.3.4: Rewritten to discuss the originator-usage period and recipient usage period of asymmetric key pairs.
24. Section 5.3.6: Further clarification of the cryptoperiod added to the Private signature key (footnote), Public signature verification key, Private authentication key (footnote), Public authentication key (footnote), Symmetric authentication key, Symmetric key-agreement key, Symmetric key-wrapping key, Symmetric RBG keys, Public key-transport key, and Private static key-agreement key.

    Corrected Symmetric data-encryption key and Symmetric key-wrapping key to agree with Table 1.

Table 1: Modified the header to refer to the originator-usage period and the recipient-usage period. Added a note to the Symmetric key-agreement key for clarification.

25. Section 5.4.2: Additional information inserted about obtaining assurance of domain parameter validity.

26. Section 5.4.3: Additional information inserted about obtaining assurance of public key validity.

27. Section 5.4.4: The details about obtaining assurance of private key possession have been removed, since this is discussed in SP 800-89. A note was added that this assurance could be obtained by a CA.

28. Section 5.5: Unnecessary text has been removed.

29. Section 5.6.1: The security-strength discussion has been revised, and a reference to SP 800-158 has been inserted.

    Deleted a note about the block size that was unnecessary.

    Table 2 has been revised to provide a visual indication of which key sizes are no longer approved for applying cryptographic protection, which are approved, and which are approved, but not specifically mentioned in the FIPS standards. The note about SHA-1 was modified.

    Table 3 and the following text have been revised to clearly indicate that SHA-1 is no longer approved for generating digital signatures. The SHA-3 hash functions are now included in the table. A note has been added to the header for HMAC.

30. Section 5.6.2: Table 4 has been updated to indicate the currently projected security strength time frames.

31. Section 5.6.3: A reference to SP 800-158 has been inserted for discussions about determining the actual security strength of a key, based on how it was generated and subsequently handled.

32. Section 6.1: Changes have been made to the integrity and confidentiality protection topics to be consistent with [SP 800-152]. For the integrity protection topic, " integrity protection can be provided by cryptographic integrity mechanisms..." has been changed to " integrity protection **shall** be provided by cryptographic integrity mechanisms...".

33. Section 6.2: An "in use" state has been introduced, along with an acknowledgement that the key may also be in transit and/or in storage.

34. Section 6.2.1.3: additional guidance has been added about the generation of the key components.

36. Section 6.2.2.3: Addition text was inserted to address the [FIPS 140-2] security level in accordance with [SP 800-152].

37. Section 6.2.3.1: A key's history has been inserted as a possible metadata item. A reference to SP 800-158 has been included to provide guidance on handling metadata.

38. Section 7 has been completely rewritten, including adding a suspended state and providing clarity on the transitions of the different key types. A suspended state has been added to Figure 3 and the discussion.

39. Section 8: The suspended state has been added to the discussions and included in Figure 5.

40. Section 8.1.5: A reference to SP 800-133 has been included.

41. Section 8.1.5.1: A sentence has been added to the end of paragraph 2 about distributing keying material to an organization's sub-entities.

42. Section 8.1.5.1.1.1: The section has been revised to clearly and more correctly describe what a trust anchor is (i.e., a CA, not a certificate for that CA).

43. Section 8.1.5.1.2: A reference to SP 800-56B has been removed, since it does not include schemes that use ephemeral keys.

44. Section 8.1.5.2, 8.1.5.2.2, and 8.2.3.2: References to the use of key update as an approved method for key change have been removed or modified.

45. Section 8.1.5.2.2.2: References to SP 800-38F, SP 800-56A and SP 800-56B have been added. A note has been added to mention authenticated encryption modes.

46. Section 8.1.5.2.3: Mentions of key wrapping have been removed, since it is not used in key-agreement schemes.

47. Section 8.1.5.3.4 has been rewritten.

48. Sections 8.2.1.1 and 8.2.1.2 : The mention of a "device" has been removed, as the appropriate reference is to cryptographic modules.

49. Section 8.2.3.2: Key update is now disallowed, as stated in SP 800-152.

50. Section 8.3.1: More guidance has been provided on using archives.

51. Section 8.3.4: The text was modified to discuss the destruction of a key, rather than the destruction of the media containing a destroyed key.

52. Section 8.3.5, paragraph 6: "...the corresponding public-key certificate **should** be revoked " has been changed to "...the corresponding public-key certificate **shall** be revoked as soon as possible," and more guidance has been provided about using revoked certificates.

53. Section 10: A reference has been included to SP 800-130 and SP 800-152.

54. Section 10.2.7: A reference to identity-based privileging has been added.

55. Appendix B.3: The first list of decision items has been replaced with a reference to Section 8.2.2.2 to avoid duplication.

56. Appendix B.3.3.1: The first sentence has been rewritten verify the edentity of the entity...", rather than "verify the authenticity...".

57. Appendix B.3.3.2: Rewritten.

58. Appendix B.3.4 and B.3.5: Text about the security strength has been removed as being inappropriate for this section.

59. Appendix C: The references have been updated, including the addition of FIPS 202, SP 800-38G, SP 800-90, SP 800-130 and SP 800-152.