Key Management Guideline

Workshop Document

November 1-2, 2001

This document has been developed specifically for the key management workshop. It is not intended as the draft of a finished document and should not be reviewed as such. The reviewer should read this workshop document with an eye to determining whether or not the concepts are correct and all necessary topics have been addressed, or it appears that they will be addressed in the future. This workshop document is being developed simultaneously with a key establishment schemes document [FIPS XXX] that will also be discussed at the workshop.

The reviewer's opinion is solicited on the basic approach taken for the final guideline, and the concepts and details addressed. Are they correct and complete? If not, the reviewer is requested to provide comments to assist in the development of the guideline. In addition, questions for the reviewer have been inserted in red.

Table of Contents

1. Introduction

Cryptographic mechanisms are one of the strongest ways to provide security services for electronic applications and protocols. Since NIST published DES in 1977, a suite of approved algorithms for unclassified but sensitive applications has been growing. New classes of algorithms have been added, such as secure hash algorithms and asymmetric algorithms for digital signatures. The suite of algorithms now provides different levels of cryptographic strength through a variety of key sizes. In addition, the algorithms may be combined in many ways to support increasingly complex protocols and applications.

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 workshop document provides background information and establishes frameworks to support appropriate decisions when selecting and using cryptographic mechanisms.

1.2 Audience

There are three primary audiences for this document: cryptographic module developers, protocol developers, and system or application owners. Cryptographic module developers may benefit from this document through a greater understanding of features required to support the intended range of applications. Protocol developers may identify appropriate suites of algorithms and gain a greater understanding of the security services provided by those algorithms. System or application owners may use this document to determine which configuration settings are most appropriate for their information.

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. [SP800-21] includes a brief introduction to cryptography. [SP 800-5] and [IPKI] provide an introduction to public key infrastructure. A mathematical review of cryptography and cryptographic algorithms may be found in [HAC]. A more accessible review of cryptography may be found in [AC].

1.3 Scope

The scope of this workshop document encompasses cryptographic algorithms, infrastructures, protocols, and applications. All cryptographic algorithms currently approved by NIST for the protection of unclassified but sensitive information are in scope, as well as algorithms (e.g., Diffie-Hellman) actively under consideration. Cryptographic infrastructures that are commonly used to distribute keys for approved algorithms are considered, such as Kerberos and the X.509 public key infrastructure. The workshop document limits the consideration of protocols and applications to those widely used by Federal agencies, such as SSL.

This workshop document 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 document. Some of the topics noted above are addressed in other NIST standards and guidance. For example public key infrastructure is addressed in [IPKI]. This workshop document should be considered as a supplement to those more focused guidelines and standards.

Finally, this workshop document does not address implementation details for cryptographic modules that may be used to achieve the security requirements identified. These details are addressed in [SP 800-21], [FIPS 140-2], and [derived test requirements].

1.4. Security Services

Cryptography may be used to perform several basic security services: confidentiality, data integrity, authentication, non-repudiation. These services may also be required to protect cryptographic keys.

1.4.1 Confidentiality

Confidentiality is a service that is used to prevent the disclosure of information to unauthorized entities. Secrecy and privacy are terms that are often used synonymously with confidentiality.

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

1.4.2 Data Integrity

Data integrity is concerned with the unauthorized alteration of information (data). This includes the insertion of additional data, and the deletion or modification of all or part of the data. Absolute protection against alteration is not possible. However, it is possible to determine (with a very high probability) whether or not data has been altered (either accidentally or deliberately). Therefore, the goal of a data integrity service is to verify that data has not been altered.

Several cryptographic mechanisms may be used to provide data integrity: digital signatures, message authentication codes and keyed hash values.

1.4.3 Authentication

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

1.4.4 Non-repudiation

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

1.5 Content / Organization

The document is organized so that each section builds upon earlier material. Frameworks and classifications established in the background material are essential to discussions of specific protocols and applications later in the document.

In Section 2, *Glossary of Terms and Acronyms*, terms and acronyms that are used in this document are defined as a reference.

Section 3, *Cryptographic Algorithms and Keys*, provides essential background material. This section provides a framework for later subjects by establishing classes of cryptographic algorithms, describing the types of functions that may be implemented using these classes, and classifying keys according to their function. As part of the classification of keys, protection requirements are identified for different types of keys. These protection requirements should be of particular interest to cryptographic module vendors and application implementers.

Section 4, *Key Management Lifecycle*, establishes a framework for the lifecycle of a cryptographic key. This section should be of particular interest to cryptographic module vendors and developers of cryptographic infrastructure services (e.g., Kerberos or public key infrastructure).

Section 5, *General Key Management Guidance*, provides guidance in several crucial areas: selection of key management policy, cryptographic algorithm and key size selection, and key establishment schemes. Key management policy includes key usage, key lifetime, parameter validation, accountability, audit, and key recovery issues. Key usage focuses on the appropriateness of using keys for different services (e.g., for both key transport and digital signatures). The key lifetime guidance describes how to determine when keys should be destroyed. Accountability and audit discusses which events should be audited and how to best implement auditing. The guidance for cryptographic key size selection establishes rough equivalencies for key sizes in different algorithms (e.g., 3096-bit DSA and 128 bit AES), criteria for establishing algorithm suites where protocols or applications use different classes in conjunction, and when to transition to new key sizes.

Section 6, *Selected Infrastructures*, is intended to provide guidance on key management issues associated with widely deployed key distribution infrastructures. This section has not yet been drafted, but subsections on public key infrastructure, Kerberos, and X9.17-style key distribution are currently planned. These subsections will address key management requirements for both the infrastructure and its users. This section should be of particular interest to parties developing or deploying infrastructures for key distribution. System and application owners may use this section to determine whether a particular infrastructure supports the security services required for their information.

Section 7, *Selected Protocols*, is intended to provide guidance in the use of common cryptographic protocols. This section has not yet been completed, but contains a preliminary discussion of TLS/SSL, and a section on S/MIME secure electronic mail is currently planned. Section 7.1, *S/MIME*, will establish key management requirements for secure electronic mail using S/MIME versions 2 and 3. Section 7.2, *TLS/SSL*, will establish key management requirements for secure web services. Developers may use this section to identify important implementation details. System and application owners may use this section to establish configuration parameters appropriate for their information.

Section 8, *Selected Applications*, is intended to provide guidance on the use of cryptographic algorithms in single user applications. One example of such an application is encrypted file storage. This section will provide guidance on the functionality and limitations of encrypted file storage. System owners can use this text to select products and configure them appropriately.

Appendices will be provided to supplement the main text where a topic demands a more detailed treatment. For example, an appendix that examines cryptographic periods for signing key pairs will be provided. This text will supplement the general guidance on cryptoperiods found in Section 5. The complexity of cryptoperiods for signing key pairs demands a more detailed treatment than for other classes of keys, so this topic will be addressed separately.

2. Glossary of Terms and Acronyms

[Note: The contents of this section are provided to assist in understanding the rest of this document. It is not the intended that the definitions be discussed at the workshop. However, written comments will be accepted at kmscomments@nist.gov.]

2.1 Glossary of Terms

Access authority	An entity responsible for monitoring and granting access privileges for other authorized entities.
Access control	Restricts access to resources only to privileged entities.
Accountability	A process that ensures that the actions of an entity may be traced uniquely to that entity.
Approved	FIPS-Approved and/or NIST-recommended. An algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, or 2) adopted in a FIPS or NIST Recommendation and specified either in an appendix to the FIPS or NIST Recommendation, or in a document referenced by the FIPS or NIST Recommendation.
Archive	See Key management archive.
Asymmetric key algorithm	See Public key cryptographic algorithm.
Attribute authority	An entity with a signing key and the authority to create attribute certificates that bind a privilege to another certificate, usually an identity certificate (i.e., a public key certificate that binds a public key with the identity of the owner of the public key).
Authentication code	A cryptographic checksum based on an Approved security function (also known as a Message Authentication Code).
Authorization	Access privileges granted to an entity; conveys an “official” sanction for an entity to access or modify privileged information.
Availability	Timely, reliable access to information by authorized entities.
Backup	Copy of information to facilitate recovery, if necessary.
Certificate	See public key certificate.
Certification authority	The entity in a Public Key Infrastructure (PKI) that is responsible for issuing user certificates, and exacting compliance to a PKI policy.
Ciphertext	Data in its encrypted form.
Compromise	The unauthorized disclosure, modification, substitution or use of sensitive data (e.g., keying material).
Confidentiality	The property that sensitive information is not disclosed to unauthorized entities.

Contingency plan	A plan that is maintained for disaster response, backup operations, and post-disaster recovery to ensure the availability of critical resources and to facilitate the continuity of operations in an emergency situation.
Cryptanalysis	1. Operations performed in defeating cryptographic protection without an initial knowledge of the key employed in providing the protection. 2. The study of mathematical techniques for attempting to defeat cryptographic techniques and information system security. This includes the process of looking for errors or weaknesses in the implementation of an algorithm or of the algorithm itself.
Cryptographic key (key)	A parameter used in conjunction with a cryptographic algorithm that determines the transformation of plaintext data into ciphertext data, • the transformation of ciphertext data into plaintext data, a digital signature computed from data, • the verification of a digital signature computed from data, • an authentication code computed from data, • a shared secret that is used to derive keying material.
Cryptographic key component (key component)	One of at least two parameters that have the same format as a cryptographic key; parameters with the same format are combined in an Approved security function to form a plaintext cryptographic key.
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 key, Data encrypting key	A cryptographic key that is used to cryptographically protect data (e.g., encrypt, decrypt, authenticate).
Data integrity	A property whereby data has not been altered in an unauthorized manner since it was created, transmitted or stored.
Data origin authentication	Corroborating that the source of the data is as claimed.
Decryption	The process of changing ciphertext into plaintext.
Digital signature	The result of a cryptographic transformation of data that, when properly implemented, provides the services of: 1. origin authentication 2. data integrity, and 3. signer non-repudiation.
Dual control	A process that uses two or more separate entities (usually persons)

	operating in concert to protect sensitive functions or information. No single entity is able to access or use the materials, e.g., cryptographic keys.
Encrypted key	A cryptographic key that has been encrypted using an Approved security function with a key encrypting key in order to disguise the value of the underlying plaintext key.
Encryption	The process of changing plaintext into ciphertext to provide confidentiality.
Entity	An individual (person), organization, device or process.
Ephemeral keys	Ephemeral keys are short-lived and are statistically unique to each execution of a key establishment process (e.g., unique to each message or session).
Hash-based message authentication code (HMAC)	A message authentication code that uses an Approved keyed hash function.
Hash function	A function that maps a bit string of arbitrary length to a fixed length bit string. Approved hash functions satisfy the following properties: 1. It is computationally infeasible to find any input that maps to any pre-specified output, and 2. It is computationally infeasible to find any two distinct inputs that map to the same output.
Hash value	The result of applying a hash function to information.
Initialization vector (IV)	A vector used in defining the starting point of an encryption process within a cryptographic algorithm.
Integrity	The property that sensitive data has not been modified or deleted in an unauthorized and undetected manner. Integrity refers to the assurance that information was not modified accidentally or deliberately during transit or storage by replacement, insertion or deletion.
Key component	See cryptographic key component.
Key compromise	A condition in which secret or private information (e.g., a key) is known by one or more unauthorized entities.
Key confirmation	A method of determining that an entity has the correct keying material.
Key de-registration	A stage in the lifecycle of keying material; the removal of all records of keying material that was registered by a registration authority.
Key encrypting key	A cryptographic key that is used for the encryption or decryption of other keys.
Key establishment	A stage in the lifecycle of keying material; the process by which cryptographic keys are securely distributed among cryptographic modules using manual transport methods (e.g., key loaders), automated

	methods (e.g., key transport and/or key agreement protocols), or a combination of automated and manual methods (consists of key transport plus key agreement).
Keying material installation	A stage in the lifecycle of keying material; the installation of keying material for operational use.
Key management	The activities involving the handling of cryptographic keys and other related security parameters (e.g., IVs and passwords) during the entire life cycle of the keys, including their generation, storage, establishment, entry and output, and destruction.
Key Management archive	A stage in the lifecycle of keying material; a repository containing keying material of historical interest.
Keying material	The data (e.g., keys and IVs) necessary to establish and maintain cryptographic keying relationships.
Key pair	A public key and its corresponding private key; a key pair is used with a public key algorithm.
Key recovery	A stage in the lifecycle of keying material; mechanisms and processes that allow authorized entities to retrieve keying material from key backup or archive.
Key registration	A stage in the lifecycle of keying material; the process of officially recording the keying material by a registration authority.
Key revocation	A stage in the lifecycle of keying material; a process whereby a notice is made available to affected entities that keying material should be removed from operational use prior to the end of the established cryptoperiod of that keying material.
Key transport	Secure transport of cryptographic keys from one cryptographic module to another module. When used in conjunction with a public key (asymmetric) algorithm, keying material is encrypted using a public key and subsequently decrypted using a private key. When used in conjunction with a symmetric algorithm, key transport is known as key wrapping.
Key update	A stage in the lifecycle of keying material; alternate storage for operational keying material during its cryptoperiod.
Key wrapping	Encrypting a symmetric key using another symmetric key (the key encrypting key). A key used for key wrapping is known as a key encrypting key.
Label	Information that either identifies an associated parameter or provides information regarding the parameter's proper protection and use.
Manual key transport	A non-electronic means of transporting cryptographic keys.
Message authentication code	Data that is associated with authenticated information that allows an entity to verify the integrity of the information. c

(MAC)	
Non-repudiation	A service that is used to provide proof of the integrity and origin of data in such a way that the integrity and origin can be verified by a third party and cannot be successfully denied by the originator.
Operational storage	A stage in the lifecycle of keying material; the normal storage of operational keying material during its cryptoperiod.
Operational use	A stage in the lifecycle of keying material; a stage whereby keying material is used for standard cryptographic purposes.
Plaintext	Intelligible data that has meaning and can be acted upon without the application of decryption.
Private key	A cryptographic key, used with a public key cryptographic algorithm, that is uniquely associated with an entity and is not made public. In an asymmetric (public) cryptosystem, the private key is associated with a public key. The private key is known only by the owner of the public/private key pair and is used to: 1. Compute the corresponding public key, 2. Compute a digital signature that may be verified by the corresponding public key, 3. Decrypt data that was encrypted by the corresponding public key, or 4. Compute a piece of common shared data, together with other information.
Pseudorandom number generator (PRNG)	A deterministic RNG that consists of an algorithm that produces a sequence of bits from an initial value called a seed. The output of the PRNG “appears” to be random, i.e., the output is statistically random and essentially unpredictable, given that the seed is not known.
Public key	A cryptographic key used with a public key cryptographic algorithm that is uniquely associated with an entity and that may be made public. In an asymmetric (public) cryptosystem, the public key is associated with a private key. The public key may be known by anyone and is used to: 1. Verify a digital signature that is signed by the corresponding private key, 2. Encrypt data that may be decrypted by the corresponding private key, or 3. Compute a piece of shared data.
Public key certificate	A set of data that uniquely identifies an entity, contains the entity's public key, and is digitally signed by a trusted party, thereby binding the public key to the entity.
Public key	A cryptographic algorithm that uses two related keys, a public key and

(asymmetric) cryptographic algorithm	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 Number Generator (RNG)	Random Number Generators (RNGs) used for cryptographic applications typically produce a sequence of zero and one bits that may be combined into sub-sequences or blocks of random numbers. There are two basic classes: deterministic and non-deterministic. A deterministic RNG consists of an algorithm that produces a sequence of bits from an initial value called a seed. A non-deterministic RNG produces output that is dependent on some unpredictable physical source that is outside human control.
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 should not be made public. The use of the term “secret” in this context does not imply a classification level, but rather implies the need to protect the key from disclosure.
Secret (symmetric) key cryptographic algorithm	A cryptographic algorithm that uses a single secret key for both encryption and decryption.
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 policy	Defines the threats that a system must address and provides high level mechanisms for addressing those threats.
Security services	Mechanisms used to provide confidentiality, data integrity, authentication or non-repudiation of information.
Seed	A secret value used to initialize a pseudorandom number generator.
Shared secret	A value that is generated during a key agreement process; the shared secret is used to derive keying material.
Signature generation	Uses a digital signature algorithm and a private key to generate a digital signature on data.
Signature verification	Uses a digital signature algorithm an a public key to verify a digital signature.
Split knowledge	A process by which keying material is split into multiple components, each component being the length of the original material. Each component provides no knowledge of the original material or the other component(s). The original material is recovered by combining the components. It is not possible to determine the resulting key until all

	required components are combined.
Static keys	Static keys are relatively long-lived and is common to a number of executions of a given algorithm.
Statistically unique	For the generation of *n*-bit quantities, the probability of two values repeating is less than or equal to the probability of two *n*-bit random quantities repeating. Thus, an element chosen from a finite set of 2^n elements is said to be statistically unique if the process that governs the selection of this element provides a guarantee that for any integer $L \leq 2^n$ the probability that all of the first L selected elements are different is no smaller than the probability of this happening when the elements are drawn uniformly at random from the set.
Symmetric key	A single cryptographic key that is used with a secret (symmetric) key algorithm.
Symmetric key algorithm	See Secret key cryptographic algorithm.
System initialization	A stage in the lifecycle of keying material; setting up and configuring a system for secure operation.
Threat	Any circumstance or event with the potential to adversely impact a system through unauthorized access, destruction, disclosure, modification of data or denial of service.
Unauthorized disclosure	An event involving the exposure of information to entities not authorized access to the information.
User initialization	A stage in the lifecycle of keying material; the process whereby a user initializes its cryptographic application (e.g., installing and initializing software and hardware).
User registration	A stage in the lifecycle of keying material; a process whereby an entity becomes a member of a security domain.

2.2 Acronyms

The following abbreviations and acronyms are used in this standard:

AES	Advanced Encryption Standard specified in FIPS 197.
ANSI	American National Standards Institute
CA	Certification Authority
DSA	Digital Signature Algorithm specified in FIPS 186-2.
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard.
HMAC	Keyed-Hash Message Authentication Code specified in FIPS 198.
IV	Initialization Vector.

MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
PKI	Public Key Infrastructure
PRNG	Pseudorandom Number Generator
RNG	Random Number Generator
TDES	Triple Data Encryption Standard; Triple DES

3. Cryptographic Algorithms, Keys, and other Keying Material

This section describes Approved cryptographic algorithms that make use of cryptographic keys to provide security services such as confidentiality, data integrity, authentication, and non-repudiation. A general discussion on keys and other keying material follows the overview of cryptographic algorithms.

3.1 Classes of Cryptographic Algorithms

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

Hash algorithms (i.e., hash functions) require no keys. Hash algorithms generate a small message digest from a large message and are used as components in many cryptographic processes, such as Message Authentication Codes (MACs) (Section 3.2.3), digital signatures (Section 3.2.4), key establishment (Section 3.2.5), and random number generation (Section 3.2.6)).

Symmetric key algorithms (sometimes known as a secret key algorithms) use a single key to transform data. Symmetric keys are often known by more than one entity; however, the key must remain secret between any entities authorized to access data protected by that algorithm and key. Symmetric key encryption algorithms are used:

- To provide data confidentiality (Section 3.2.2) - the same key is used to encrypt and decrypt data. Two symmetric key encryption algorithms are approved for Federal Government use: the Advanced Encryption Standard (AES), defined in FIPS 197, and The Triple Data Encryption Standard (TDES), adopted in FIPS 46-3[1].
- To provide authentication and integrity services (Section 3.2.2.1) 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 hash function as their cryptographic primitive.
- As part of the key establishment process (Section 3.2.5).
- To generate pseudorandom numbers (Section 3.2.6).

Asymmetric key algorithms, commonly known as public key algorithms, use two related keys (i.e., a key pair) to perform their functions: a public key and a private key. The public key may be known by anyone; the private key must be under the sole control of the entity that “owns” the key pair. Even though the public and private keys of a key pair are related, knowledge of the public key does not reveal the private key. Public key algorithms are commonly used in the computation of digital signatures (Section 3.2.4) and in the establishment of cryptographic keying material (Section 3.2.5).

[1] FIPS 46-3 adopts ANSI X9.52, a standard developed by the American National Standards Institute (ANSI). ANSI X9.52 specifies the TDES and its modes of operation.

3.2 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.

3.2.1 Hash Function

Hash functions are used with other algorithms to provide a number of security services. A hash function may be used in conjunction with a digital signature algorithm to compute a digital signature (see FIPS 186-2). A hash function may be used as part of a random number generator. A hash function may be used with a key as part of its input to provide a Keyed-Hash Message Authentication Code (HMAC) (see Section 3.2.3.2 and FIPS 198). Approved hash functions are defined in FIPS 180-2.

A hash function takes an input of arbitrary length and outputs a smaller fixed size value. Common names for the output of a hash function are hash value, hash, message digest, and digital fingerprint. The maximum number of input bits is determined by the design of the hash function, as is the size of the output. All Approved hash functions are cryptographic hash functions. Four hash functions will be approved for Federal Government use and are defined in FIPS 180-2: SHA-1, SHA-256, SHA-384 and SHA-512. The hash size produced by SHA-1 is 160 bits, 256 bits for SHA-256, 384 bits for SHA-384, and 512 bits for SHA-512.

3.2.2 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 to plaintext using decryption. The approved algorithms for encryption/decryption are symmetric key algorithms: AES and TDES. Each of these algorithms operates on blocks (chunks) of data during a single encrypt or decrypt operation. For this reason, these algorithms are commonly referred to as block cipher algorithms.

3.2.2.1 Advanced Encryption Standard (AES)

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

3.2.2.2 Triple DES (TDES)

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

3.2.2.3 Modes of Operation

With a block cipher algorithm, the same plaintext block will always encrypt to the same ciphertext block whenever the same key is used. This is a disadvantage when the information is longer than a block in length. Cryptographic modes have been defined to alleviate this problem. NIST Recommendation XXX defines modes of operation for the encryption and decryption of data using block cipher algorithms such as AES and TDES. ANSI X9.52 (adopted by FIPS 46-3) defines three additional TDES modes that pipeline or interleave the cryptographic operations to improve the efficiency of TDES.

3.2.3 Message Authentication Codes (MACs)

Message Authentication Codes (MACs) provide data authentication and integrity. A MAC is a cryptographic checksum on the data that is used to provide assurance that the data has not changed and that the MAC was computed by the expected entity. The computation of a MAC requires the use of a secret key that is known only by the party that generates the MAC and by the intended recipient(s) of the MAC and the data on which the MAC is calculated. Two types of algorithms for computing a MAC have been approved: MAC algorithms that are based on block cipher algorithms, and MAC algorithms that are based on hash algorithms.

3.2.3.1 MACs Using Block Cipher Algorithms

NIST Recommendation XXX defines modes to compute a MAC using Approved block cipher algorithms such as AES and TDES. The key and block size used to compute the MAC depends on the algorithm used. A CBC-MAC is computed using the CBC mode and the selected block cipher algorithm in a single pass through the entire data. Encryption or decryption of the data is not performed in the same process. [Comment: The definition of the modes is a multi-step process. An initial modes recommendation will soon be available. NIST is considering additional modes that would provide authentication and encryption in the same process.]

3.2.3.2 MACs Using Hash Functions

FIPS 198, *Keyed-Hash Message Authentication Code (HMAC)*, specifies the computation of a MAC using an Approved hash function. The algorithm requires a single pass through the entire data. A variety of key sizes are allowed for HMAC; the choice of key size depends on the amount of security to be provided to the data. See Section 5.2 for guidance in the selection of key sizes.

3.2.4 Digital Signature Algorithms

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

3.2.4.1 DSA

The Digital Signature Algorithm (DSA) is specified in FIPS 186-2 and will be revised as FIPS 186-3 in the near future. The revised standard will define this algorithm for specific key sizes[2]

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

between 1024 and 15,360 bits. The revised DSA will produce digital signatures[3] between 320 and 1024 bits, depending on the key size.

3.2.4.2 RSA

The RSA algorithm, as specified in ANSI X9.31 and PKCS #1 (version 1.5 and higher), has been adopted for the computation of digital signatures in FIPS 186-2. ANSI X9.31 defines key sizes[4] of 1024 bits or more in increments of 256 bits; PKCS #1 does not specify key sizes. An additional difference between ANSI X9.31 and PKCS #1 is that each specifies a different format for constructing the data to be signed. RSA produces digital signatures that are one bit in length less than the key size.

3.2.4.3 ECDSA

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

3.2.5 Key Establishment Algorithms

Key establishment algorithms are used to set up keys to be used between communicating entities. Two types of key establishment have been defined: key transport and key agreement. Approved key establishment schemes are provided in FIPS XXX.

Key transport is the distribution of a key (and other keying material) from one entity to another entity. The keying material is usually encrypted by the sending entity and decrypted by the receiving entity(ies). If a symmetric algorithm (e.g., AES) is used to encrypt the keying material to be distributed, the sending and receiving entities must know the secret key encrypting key. In this case, the key transport algorithm is commonly known as a key wrapping algorithm. If a public key algorithm is used to distribute the keying material, the key encrypting key is actually a key pair. In this case, the sending entity encrypts the keying material using the receiving entity's public key; the receiving entity decrypts the received keying material using the associated private key.

Key agreement is the participation by both entities (i.e., the sending and receiving entities) in the creation of shared keying material. This is accomplished using public key techniques. Each entity has either a static key pair or an ephemeral key pair or both.

FIPS XXX adopts selected key establishment schemes defined in ANSI standards that use public key algorithms: ANSI X9.42, X9.44, and ANSI X9.63. ANSI X9.42 specifies key agreement schemes, ANSI X9.44 specifies key transport schemes, and ANSI X9.63 specifies both key agreement and key transport schemes. FIPSXXX also specifies a technique for key wrapping.

---

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

[4] For RSA, the key size is considered to be the size of the modulus *n*.

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

3.2.5.1 Discrete Log Key Agreement Schemes

Key agreement schemes based on the intractability of the discrete logarithm problem and using finite field arithmetic have been adopted in FIPS XXX from ANSI X9.42. FIPS XXX has adopted seven of the eight schemes defined in ANSI X9.42. Each scheme provides a different configuration of required key pairs, depending on the requirements of a communication situation.

3.2.5.2 RSA Key Transport

RSA key transport schemes have been adopted from ANSI X9.44. [Comment: Further text to be developed, depending on the schemes document]

3.2.5.3 Elliptic Curve Key Agreement and Key Transport

Elliptic curve key agreement and key transport schemes based on the intractability of the discrete logarithm problem and using elliptic curve arithmetic have been adopted from ANSI X9.63 in FIPS XXX. Seven of the eleven key agreement schemes have been adopted. [Comment: Further text will be added after the schemes working group has addressed key transport schemes.]

3.2.5.4 Key Wrapping

Key wrapping is the encryption of a key by a key encryption key using a symmetric algorithm (e.g., an AES key is encrypted by an AES key encrypting key). [Comment: Further text will be added after the key wrapping scheme to be included in the schemes document is available.]

3.2.6 Random Number Generation

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

Note: non-deterministic RNGs are often called true RNGs or hardware RNGs. A non-deterministic RNG need not consist only of hardware.

Pseudorandom number generation (PRNG) algorithms defined in FIPS 186-2 (including those defined in ANSI X9.31 and ANSI X9.62[6]) use either hash functions or AES to generate random numbers that may be used for cryptographic applications (e.g., key or IV generation). PRNGs are initialized with a starting value called a seed and, in the case of the PRNG defined using AES, requires a symmetric key.

3.3 Cryptographic Keys and Other Keying Material

Several different classes of keys are used by the Approved algorithms. Many of these keys are associated with other keying material. Each requires various degrees of protection.

[6] As allowed in FIPS 186-2.

3.3.1 Classes of Keys and Protection Requirements

Several different classes of keys, grouped according to function and according to their useful life span (life cycle), are defined. The life cycle of each type of key is further discussed in Section 4. Table 1 provides a summary of the protection requirements for these keys during distribution and storage. Methods for providing the necessary protection are discussed in Section 4.

Guide to Table 1:

- Column 1 identifies the key types.
- An **X** in columns 2 and 3 identifies the keying material that requires confidentiality and integrity protection.
- An **X** in column 4 (long term availability) indicates keying material that may need to be saved for later recovery (depending on the application) if the normally available copy of the keying material is lost or corrupted.
- An **X** in column 5 (Associated with usage or application) indicates that the association with a given cryptographic mechanism (e.g., the usage of the key for digital signatures or key establishment) or with a particular application must be provided to ensure that the keying material is not used incorrectly.
- An **X** in column 6 (association with owner or other entity) indicates that the keying material must be correctly associated with another entity.
- Column 7 indicates other information that should be "linked" or "bound" to the keying material.
- Column 8 indicates which keying material needs to be validated as defined in FIPS XXX and in this workshop document.

Key Types:

[Note: We have tried to include a comprehensive list of key types. However, in some cases, we aren't sure if such keys are even used? Are there key types other than those listed?]

1. *Signing keys*: Signing keys are the private keys of a public/private key pair that are used by public key algorithms to generate digital signatures with possible long-term implications. When properly handled, signing keys can be used to assure authentication, integrity and non-repudiation. Signing keys require confidentiality and integrity protection, and correct association with any domain parameters[7]. If multiple signing keys are used (e.g., for different applications), then provision must be made to ensure that each key is only used for the appropriate application or usage[8].

[7] DSA and ECDSA require domain parameters, whereas RSA has no such parameters.

[8] Key usage refers to the function for which the key is used (e.g., digital signatures, key establishment, encryption). See Section 3.2.

2. *Signature verification keys*: Signature verification keys are the public keys of a public/private key pair that are used by a public key algorithm to verify digital signatures, either for non-repudiation purposes, to determine the integrity of data, to authenticate a user's identity, or a combination thereof. The integrity of these keys must be protected, and an association with any domain parameters, the usage or application, the public key owner and with the correct signing key must be assured. The key must be validated to ensure that the purported owner of the key has the private signing key (i.e., proof of possession must be established). The verification key needs to be available while any data signed using the associated private key may need to be verified. There is no requirement for the confidentiality of signature verification keys.
3. *Secret authentication keys*: Secret authentication keys are used with symmetric key algorithms to authenticate users, messages, or communication sessions. These keys must remain confidential, and the integrity must be protected. The association with another entity using that key must be maintained in order to provide entity authentication. If multiple authentication keys are used (e.g., for different applications or different communication associations or different data), then provision must be made to ensure that each key is only used for the appropriate application, communication association or data. If the associated protected data is to remain available, and its authenticity is to remain verifiable for an extended period of time, the authentication key must also remain available for that period.
4. *Private authentication keys*: Private authentication keys are used with public key algorithms to authenticate users, messages, or communication sessions. Non-repudiation is not necessary for private keys used only for authentication. However, these keys must remain confidential, and the integrity must be protected. If multiple authentication keys are used (e.g., for different applications or different communication associations or different data), then provision must be made to ensure that each key is only used for the appropriate application.
5. *Public authentication keys*: Public authentication keys are used with public key algorithms to authenticate users, messages, or communication sessions. The integrity, but not the confidentiality, of these keys must be protected. The association with the public key's owner must be maintained in order to provide entity authentication. If multiple authentication keys are used (e.g., for different applications or different communication associations or different data), then provision must be made to ensure that each key is associated with the appropriate application, communication association or data. If the associated protected data is to remain available, and its authenticity is to remain detectable for an extended period of time, the authentication key must also remain available for that period.
6. *Long term data encryption keys*: These keys are symmetric (secret) keys that are used with symmetric algorithms to protect the confidentiality of data over long periods. Keys used for the encryption of other keys are discussed below. These keys require confidentiality and integrity protection, and must remain available and associated with the encrypted data, communication association or application as long as the data encrypted under these keys is maintained in its encrypted form.
7. *Short term data encryption keys:* These keys are symmetric (secret) keys that are used with symmetric algorithms to protect the confidentiality of data over short periods, such as a communication session or a single message. Keys used for the encryption of other keys are discussed below. These data encryption keys are generated as needed. The confidentiality

and integrity of these keys must be maintained until the entire session or message has been decrypted. Once they are no longer needed, they must be securely destroyed (see Section 4.10).

8. *Random Number Generation keys*: These keys are symmetric (secret) keys used with a symmetric algorithm to generate pseudorandom numbers. These keys require confidentiality and integrity protection, should be associated with the RNG application, and should be retained until replaced or no longer needed.

9. *Key encrypting keys used for key wrapping*: Key encrypting keys used for key wrapping are used with symmetric key algorithms. This key encrypting key may encrypt either data encrypting keys or other key encrypting keys. These keys require confidentiality and integrity protection, and may need to remain available (for possible key recovery). These keys may also need to remain associated with the application, the other entity, and the keys they encrypt for the life of any key that is encrypted by the key encrypting key and the data associated with the encrypted keys (because of a possible compromise).

10. *Master key used for key derivation*: A “master key” may be used to derive other keying material (see FIPS XXX). These keys require confidentiality and integrity protection, and may need to remain available (for possible key recovery). These keys may also need to remain associated with the application, the other entity, and the keys that are derived for the life of any derived key and the data associated with the derived keys (because of a possible compromise).

11. *Keys derived from a master key*: These keys must be derived in accordance with FIPS XXX. They should be protected in accordance with their use, and may need to remain associated with the Master Key from which they are derived.

12. *Key transport private keys*: Key transport private keys are used to decrypt keys that have been encrypted by the associated public key using a public key algorithm. They are usually used to establish multiple keys (e.g., data encrypting keys or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors). These private keys require confidentiality and integrity protection. They may need to remain available (for possible key recovery), and may need to remain associated with the correct application or usage, and with the keys they decrypt (because of a possible compromise).

13. *Key transport public keys*: Key transport public keys are used to encrypt keys using a public key algorithm. They are used to establish keys (e.g., data encrypting keys or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors). These keys require integrity protection (but not confidentiality protection), and must be correctly associated with the key’s owner. These keys should be validated prior to their use, and should be retained until no longer needed (e.g., the public/private key pair is replaced, or key transport will no longer be required).

14. *Static key agreement private keys*: Static private keys used for key agreement are used to establish keys (e.g., key encrypting keys, data keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors). These private keys require confidentiality and integrity protection, and must remain available and associated with the correct domain parameters. If multiple static private keys are used (e.g., for different applications), then provision must be made to ensure that each key is only used for the appropriate application.

These keys should be retained until replaced or no longer needed to determine a key (e.g., key agreement will not be performed).

15. *Static key agreement public keys*: Static public keys used for key agreement are used to establish keys (e.g., key encrypting keys, data keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors). The keys require integrity protection, and the correct association with the owner of that key and any domain parameters must be assured. If multiple static public keys are used (e.g., for different applications), then provision must be made to ensure that each key is only used for the appropriate application. These keys should be validated prior to their use and should be retained until replaced or no longer needed to determine a key (e.g., key agreement will not be performed).

16. *Ephemeral key agreement private keys*: Ephemeral private keys used for key agreement are used once to establish one or more keys (e.g., key encrypting keys, data keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors). These keys require confidentiality and integrity protection during the key agreement process. The ephemeral private keys must be destroyed at the completion of the key agreement process (see Section 4.10).

17. *Ephemeral key agreement public keys*: Ephemeral public keys used for key agreement are used once to establish one or more keys (e.g., key encrypting keys, data keys, or MAC keys) and, optionally, other keying material (e.g., Initialization Vectors). These keys require integrity protection, but not confidentiality protection. These keys should be validated prior to their use, and should be destroyed at the completion of the key agreement process (see Section 4.10).

[Are keys used for authorization? There are none specifically identified in a FIPS, though a FIPS algorithm could be used for this purpose.]

18. *Secret authorization key*: Secret authorization keys are used to provide access privileges to an entity. The key is known by the access authority and an entity seeking access to resources. The secret authorization key requires confidentiality and integrity protection, and should be correctly associated with the usage and application, and with the entity seeking access.

19. *Private authorization key*: Private authorization keys are used to provide access privileges to an entity. The key is known only by an entity seeking access to resources. The private authorization key requires confidentiality and integrity protection, and should be correctly associated with the usage and application.

20. *Public authorization key*: Public authorization keys are used to verify access privileges by an entity that knows the associated private key. The public key may be known by anyone. The public authorization key requires integrity protection, and should be correctly associated with the usage and application, and with the entity seeking access.

3.3.2 Other Keying Material and its Protection

Other information used in conjunction with cryptographic algorithms also requires protection. Table 1 includes a summary of the protection requirements for this material during distribution and storage. Methods for providing the necessary protection are discussed in Section 4.

1. *Domain Parameters*: Domain parameters are used in conjunction with some public key algorithms to generate key pairs or to create digital signatures (i.e., DSA, and the Diffie-Hellman and MQV key agreement schemes specified in [FIPS XXX]). The domain parameters require integrity protection, must be associated with the correct application and the public/private key pairs with which they are used, and must be validated before use. They must be available until they are replaced or no longer needed to generate keying material, or generate and verify digital signatures.
2. *Initialization Vectors*: Initialization vectors (IVs) are used by several modes of operation for encryption and decryption (see Section 3.2.2.3) and for the computation of MACs using block cipher algorithms (see Section 3.2.3.1). IVs require integrity protection and must be available for the lifetime of any data protected using the IVs.
3. *Shared Secrets:* Shared secrets are generated during a key establishment process as defined in FIPS XXX. These shared secrets require confidentiality and integrity protection; must be associated with the appropriate application, usage or other party; and may be needed to be available on a long term basis and be associated with data or keys that are protected by the shared secret (e.g., for possible key recovery). The shared secrets must be available until they are no longer needed, at which time they must be securely destroyed (see Section 4.9).
4. *Seeds*: Seeds are used in the generation of *pseudorandom* numbers. Usually, the confidentiality of seeds must be protected. However, for some applications (e.g., elliptic curve key agreement schemes), seeds may be provided to allow a party to verify that domain parameters were generated properly. In this case, the association of the seeds to the application would need to be assured, and would need to be retained until such verification was no longer required.
5. *Intermediate Results*: The intermediate results of cryptographic operations must remain confidential and must be destroyed as soon as no longer needed (see Section 4.10). Intermediate results should be associated with the application or process that created the intermediate result.

Table 1 : Protection Requirements for Key Classes

	Confiden-tiality	Integrity	Long Term Availability	Associated with usage or application	Association with owner/other entity	Associated with other info.	Validation
Signing keys	X	X		X		Domain parameters; signature verification key	
Signature verification keys		X	X	X	X	Domain parameters; signing key	For association with private key
Secret authentication keys	X	X	X	X	X	Authenticated data	
Private authentication key	X	X		X		Public authentication key	
Public authentication key		X	X	X	X	Authenticated data; private authentication key	For association with private key
Long term data encryption keys	X	X	X	X	X	Encrypted data	
Short term data encryption keys	X	X					
RNG keys	X	X		X			
Key encrypting key used for key wrapping	X	X	X	X	X	Encrypted keys	

Master key used for key derivation	X	X	X	X	X	Derived keys	
Keys derived from a Master Key	X?	X	X?	X?	X?	Master key and protected data	
Key transport private keys	X	X		X?		Encrypted keys; key transport public key	
Key transport public keys		X	X		X	Key transport private key	X
Static key agreement private keys	X	X	X	X?		Domain parameters; static key agreement public key	
Static key agreement public keys		X	X	X?	X	Domain parameters; static key agreement private key	X
Ephemeral key agreement private keys	X	X					
Ephemeral key agreement public keys		X					X
Secret authorization key	X	X		X	X		
Private authorization key	X	X		X		Public authorization key	
Public authorization key		X		X	X	Private authorization key	
Domain parameters		X	X?	X		Private and public keys	X
Initialization vectors	?	X	X			Protected data	
Shared secrets	X	X	?	X	X	?	

Seeds	X?			X?		Generated data?	
Intermediate results	X			X		Process data	

4. Key Management Lifecycle

Cryptographic key management encompasses the entire life cycle of cryptographic keys and other keying material. Basic key management guidance is provided in [SP800-21].

A single item of keying material (e.g., a key) has several states during its life, though some of these states may, in fact, be very short:

- Pre-operational: The keying material is not yet available for normal cryptographic operations.
- Operational: The keying material is available and in normal use.
- Post-operational: The keying material is no longer in normal use. but access to the material is possible.
- Obsolete/destroyed: The keying material is no longer available. All records of its existence have been deleted.

The following subsections discuss the various stages of key management for a given entity.

- User registration
- System and user initialization
- Keying material installation
- Key establishment
- Key registration
- Operational use
- Operational storage
- Key backup
- Key archive
- Key recovery
- Key de-registration and destruction
- Key revocation

[Question: Do we need all of the following stages? The list was based on [HAC]. A diagram visualizing the relationships between the stages will be inserted later.]

4.1 User Registration

During user registration, an entity becomes an authorized member of a security domain. This includes the acquisition,, or creation and exchange of initial keying material. [Note: Guidance needs to be provided. The guidance could be provided to include user authentication and the presentation of credentials to establish identity]

4.2 System and User Initialization

System initialization involves setting up/configuring a system for secure operation. [Note: Guidance could include topics on user IDs, passwords, and the validation of system users.]

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 4.3) of the initial keying material obtained during user registration. [Note: The guidance could include the installation of a key at a CA, trust parameters, policies, trusted parties, and algorithm preferences.]

4.3 Keying Material Installation

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

Note: This section will contain guidance on:

- the initial installation of keying material (e.g., by manual entry, electronic key loader, by a vendor during manufacture), addressing the protection of the keying material during entry into a software/hardware/system/application/cryptomodule/ device, and taking into account the requirements of FIPS 140-2, its differing requirements based on levels of protection and additional procedures that may be required,
- issues related to the installation of additional keying material,
- issues related to replacing existing keying material that are not covered in Section 4.8,
- issues related to keying material installation during key recovery that are not addressed in Sections 4.9 or 5.1.7.

Many applications/systems are provided by the manufacturer with keying material that may be used to test that the newly installed application/system is functioning properly. This test keying material must be replaced prior to operational use by installing initial operational keying material.

4.4 Key Establishment

Key establishment includes the generation and sharing of cryptographic keys and other keying material between entities.

4.4.1 Generation and Distribution of Public/Private Key Pairs

Public/private key pairs are used with digital signature and key establishment algorithms (see Sections 3.2.4 and 3.2.5). They should be generated in accordance with the mathematical specifications of the appropriate Approved standard. It is recommended that the key pairs be generated within a FIPS 140-2 validated cryptographic module [FIPS140-2]. Alternatively, the key pair generation process may be performed in a facility to which access is controlled, and access to plaintext private keys is prevented.

Private signing, authentication and authorization keys should not be distributed to other entities.

The distribution of public keys and of private keys other than signing, authentication and authorization keys is dependent on the type of key and whether it is static or ephemeral. The keys should be protected in accordance with Section 3.3.1.

4.4.1.1 Distribution of Static Public Keys

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

a) the true owner of the key is known (i.e., the owner of the public/private key pair), [Note: This does not take into account the use of anonymous public keys.]

b) the purpose/usage of the key is known (e.g., RSA digital signatures or elliptic curve key agreement),

c) any parameters associated with the public key are known (e.g., domain parameters), and

d) the public key has been properly generated (e.g., it's mathematical properties are correct (public key validation) and/or the owner of the public key actually has the associated private key).

For example, a public key may be received in a public key certificate signed by a trusted CA. Services in the above list that are not provided by the CA (e.g., public key validation) should be performed by the receiver of the public key prior to the use of the key.

Alternatively, a public key may be received directly from a known and trusted entity who is the owner of the public key in a face-to-face meeting or via a trusted courier (manual distribution). Assurances in the above list that are not provided by the public key owner (e.g., public key validation and determination that the public key owner has the associated private key) should be acquired by the receiver of the public key prior to use of the key.

Keys falling into this category are:

- The *signature verification key,*
- The *public authentication key,*
- The *key transport public key,*
- The *static key agreement public key,* and
- The *public authorization key*.

In the case of a signature verification key and its associated private key, the owner should generate the keying material rather than any other entity; this will allow non-repudiation.

4.4.1.2 Distribution of Ephemeral Public Keys

Ephemeral keys are short-lived and are statistically unique to each execution of a key establishment process (e.g., unique to each message or session). Ephemeral public keys and the associated private keys should be generated in accordance with an Approved key establishment scheme [FIPSXXX].

The ephemeral public keys should be distributed using a secure key establishment protocol. The key establishment process (i.e., the key establishment scheme + the protocol + any associated negotiation) should provide a recipient with assurances a-c in Section 4.4.1.1. The recipient should perform public key validation as specified in [FIPSXXX].

4.4.1.3 Distribution of Centrally Generated Private Keys

[Note: This would include the distribution of key transport private keys and static key agreement private keys.]

4.4.2 Generation and Distribution of Symmetric Keys

Symmetric keys may be:

- generated and subsequently distributed either manually, using a public key transport mechanism, or using a previously distributed or agreed upon key encrypting key (see Sections 4.4.2.1 and 4.4.2.2), or
- determined using a key agreement scheme (i.e., the generation and distribution are accomplished with one process) (see Section 4.4.2.3).

The symmetric keys used for the encryption/decryption of data or other keys and for the computation of MACs (see Sections 3.2.2 and 3.2.3) must be determined by an Approved method. The keys should be randomly generated and (optionally) distributed (transported) to another party, or may be determined by a key agreement mechanism. The keys should be provided with protection that is consistent with Section 3.3.1.

4.4.2.1 Key Generation

Symmetric (secret) keys should be generated by an Approved key generation algorithm in a FIPS 140-2 validated cryptographic module or in a facility to which access is controlled, and access to the plaintext secret keys is prevented.

If the keys are used to protect stored information (either data or other keys), the keys should be stored in a manner that associates the keys with the protected information. If the keys are used to protect data or keys to be communicated between entities, one of the entities generates the keys and transports (sends) the keys to the other entity(ies).

4.4.2.2 Key Distribution

Keys generated in accordance with Section 4.4.2.1 as key encrypting keys (used for key wrapping) or for the protection of communicated information may be distributed manually (manual key transport) or using an electronic key transport protocol (electronic key transport).

Manual Key Distribution/Transport: Keys distributed manually (i.e., by other than an electronic key transport protocol) should be protected throughout the distribution process. The keys should either be encrypted or protected using split knowledge. The mechanism should assure:

- the authorized distribution of the keys,
- that the entity distributing the keys is trusted by both the entity that generates the keys and the entity(ies) that receives the keys,
- the keys are protected in accordance with Section 3.3.1,
- that the keys are received by the authorized recipient, and

- the confidentiality and integrity of the keys during transport.

Electronic Key Distribution/Transport: This method distributes keys via a communication channel (e.g., the Internet or a satellite transmission). Electronic key distribution/transport requires the prior distribution of other keys (i.e., key encrypting keys) that will enable the distribution of the newly generated secret/symmetric keys. The key encrypting keys may be either secret key encrypting keys used for key wrapping (see Section 3.2.5.4), or the public key of a public/private key pair (see Section 3.2.5.2). The new secret keys should be generated as specified in Section 4.4.2.1. The public key should be distributed as specified in Section 4.4.1.1. Approved schemes for the distribution/transport of keys is provided in [FIPSXXX]. The key transport scheme, together with the associated key establishment protocol should provide assurance that:

- the distributed key is not disclosed or modified,
- the keys are protected in accordance with Section 3.3.1,
- the recipient has received the correct key,
- [other attributes?]

Keys falling into this category are:

- The *secret authentication key*,
- The *long term data encryption key*,
- The *short term data encryption key*,
- The *key encrypting key used for key wrapping*,
- The *master key used for key derivation*, and
- The *secret authorization key*.

4.4.2.3 Key Agreement

Both the generation of keying material and the “distribution” of that material may be accomplished using a key agreement scheme. Key agreement is used in a communication environment to establish keys using information contributed by all parties in the communication (most commonly, only two parties). Approved key agreement schemes are provided in [FIPSXXX]. Key agreement requires the availability of public/private key pairs that will enable the derivation of secret keys and other keying material (e.g., IVs). A given scheme may use either static or ephemeral key pairs or both. The key pairs should be generated and distributed as discussed in Section 4.4.1. A key agreement scheme and its associated key establishment protocol should provide the following assurances:

- each entity to the key establishment knows the correct identity of the other entity(ies), [Note: This does not take anonymous entities into account. This will be addressed later.]
- the keys used in the key agreement scheme are correctly associated with the entities involved in the key establishment process,
- the public keys have been validated,
- the derived keys are correct (if key confirmation is used),

- [other attributes?]

4.4.3 Generation and Distribution of Other Keying Material

Keys are usually generated in conjunction with or are used with other keying material. This material should be protected in accordance with Section 3.3.2.

4.4.3.1 Domain Parameters

Domain parameters are used by some public key algorithms to generate public/private key pairs or to compute digital signatures. Typically, domain parameters are generated infrequently and used by a community of users for a long period of time. Domain parameters may be distributed in the same manner as the public keys with which they are associated, or may be made available at some other accessible site. The domain parameters should be validated prior to use, either by a trusted entity that indicates its “blessing” on the parameters, or by the entity’s themselves. Domain parameter validation is addressed in FIPS XXX.

4.4.3.2 Initialization Vectors

Initialization vectors are used by several modes of operation for encryption and decryption , or for authentication. The criteria for IVs is provided in [MODES]; the protections required for IVs are defined in Section 3.3.2. IVs may be distributed in the same manner as their associated keys, or may be distributed with the information that uses the IVs as part of the encryption or authentication mechanism.

4.4.3.3 Shared Secrets

Shared secrets are computed during a key agreement process and are subsequently used to derive keying material. Shared secrets are generated as specified by the appropriate key agreement scheme (see FIPS XXX), but should never be distributed.

4.4.3.4 Seeds

Seeds are used to initialize a pseudorandom number generator (PRNG). The criteria for the selection of a seed is provided in the specification of an Approved PRNG. The seeds may or may not be distributed, depending on the algorithm that uses the resulting random numbers. For example, seeds are optional in the list of domain parameters for elliptic curve techniques, and may be distributed as part of the domain parameters (see Section 4.1.1.1); in this case, the seeds may be used to validate the domain parameters. Seeds are never to be disclosed when used to generate keys.

4.4.3.5 Intermediate Results

Intermediate results occur during computation using cryptographic algorithms. These results should never be distributed.

4.5 Key Registration

During key registration, keying material is bound to information or attributes associated with a particular entity. This information typically includes the identity of the entity associated with the key material, but may also include authorization information or specify the level of trust. This step is typically performed when the entity is a participant in a key management infrastructure, such as a public key infrastructure (PKI) or Kerberos realm. The binding is performed by a trusted third party, such as a PKI certification authority or a Kerberos realm server.

When a CA performs the binding, the public key and associated attributes are placed in a public key certificate, which is digitally signed by the CA. In this case, the registered keying material may be publicly available. When a Kerberos realm server performs the binding, a symmetric key is stored on the server with the corresponding attributes. In this case, the registered keying material is maintained in confidential storage.

[Note: Need to add guidance]

4.6 Operational Use

The objective of the key management lifecycle is to facilitate the operational availability of keying material for standard cryptographic purposes. Under normal circumstances, a key remains operational until the end of the key's cryptoperiod (i.e., the expiration date). [Note: Need to add guidance for the various types of keys and specify that the keys should be used only for their intended purposes.]

4.7 Storage of Keying Material

Keying material should be stored depending on its type, protection requirements and lifecycle stage. When the keying material is required for operational use, the keying material is acquired from operational storage when not present in active memory. If the keying material in active memory or operational storage is lost or corrupted, the keying material may be recovered from backup storage, providing that the keying material has been backed up (see Sections 4.7.3 and 4.9). After the end of a key's cryptoperiod, keying material may be recovered from archival storage, providing that the keying material has been archived (see Sections 4.7.4 and 4.9).

The material 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 operational storage. The material may be stored in electronic form on a removable media (e.g., a CD-ROM) or in hard copy form and placed in a safe; this would be typical for backup or archive storage.

Table 1 summarizes the storage requirements for each key.

4.7.1 General Protection Methods

4.7.1.1 Confidentiality

Several mechanisms may be used to provide confidentiality for keying material:

- The keying material may reside in an Approved cryptographic module (cryptomodule). The cryptomodule must be designed to be compliant with FIPS 140-2 and tested by an accredited laboratory of the Cryptographic Module Validation Program (CMVP). Information on this program is available at http://csrc.nist.gov/cryptval.
- The keying material may reside in an appropriately configured and Approved trusted operating system environment.
- Keying material may be stored in a secured environment (e.g., in a safe with limited access). In this case, the material must either be encrypted or stored using dual control procedures (i.e., two or more individuals must be required for access to the material).
- Keying material may be split into multiple components. Each component must be the same length as the original cryptographic material and should appear to be a random

value. The set of components must be recombined in a secure environment (e.g., in an Approved cryptomodule) when reconstructing the original material. The components must be stored separately, under dual control, split knowledge procedures. A typical method for splitting cryptographic material into two components, for example, is to generate a random bit string of the intended length of the components, and exclusive Or-ing the bit string to the cryptographic material. The components consist of the random bit string and the result of the exclusive OR operation.

4.7.1.2 Integrity

Integrity protection for data is concerned with the prevention and/or detection of modifications to the information. Absolute protection against modification is not possible. The best that can be done is to use reasonable measures to prevent modifications, and to use methods to detect (with a very high probability) any modifications that occur.

All keying material requires integrity protection. Integrity protection is provided for keying material when it resides in an Approved cryptomodule or operating system. Alternatively, the integrity of the cryptographic material may be protected by storing it in a secure environment (e.g., a safe with limited access), by creating multiple copies that are compared before use, or by using a cryptographic mechanism, such as a MAC, digital signature, or hash function.

4.7.1.3 Association with Usage or Application

Keying material is used with a given cryptographic mechanism (the usage of the key, e.g., digital signatures or key establishment) or with a particular application. Protection should be provided to ensure that the cryptographic material is not used incorrectly. This protection may be provided by separating the keying material from that of other mechanisms or applications, or by appropriate labeling of the keying material. [Note: Need to provide guidance on labeling]

4.7.1.4 Association with the Other Entity

Many cryptographic keys must be correctly associated with another entity. A symmetric (secret) key used for the encryption of information, or keys used for the computation of a MAC must be associated with the other entity(ies) that shares the key. Public keys must be correctly associated (bound) with the owner of the public/private key pair. The symmetric keys and public keys may retain their association during storage by separating the keys by “entity” or by properly labeling the keys.

4.7.1.5 Long Term Availability

Some keying material may be easily replaced without serious consequences if the material becomes unavailable (e.g., is lost or has been modified). Other keying material may need to be readily available for as long as information is protected by that keying material. The primary method for providing this protection is to make one or more copies of the keying material that are stored in separate locations (i.e., back up the keying material). Recovery of this material is discussed in Section 4.9. The availability requirements for different keys is identified in Section 3.3.1.

4.7.1.6 Association with Other Information

An association may need to be maintained between protected information and the key (or the associated key) that protected that information.

- *Signing keys* used with the DSA and ECDSA signature algorithms must remain associated with the domain parameters in order that digital signatures can be created.
- *Public keys used to verify digital signatures* must be associated with the signed information. The public key or a pointer to that public key should be stored with the signed information. If the signed information points to the pubic key, an identification/label should be stored with the public key for easy reference.
- *Secret authentication keys* must remain associated with the authenticated information during the lifetime of that information. The authentication key should be stored with the protected information, or a pointer to the authentication key should be stored with the protected information. If the authenticated information points to the secret key, an identification/label should be stored with the secret key for easy reference.
- *Public authentication keys* must remain associated with the information that was protected by the associated private authentication key during the lifetime of the protected information. The authentication key should be stored with the protected information, or a pointer to the authentication key should be stored with the protected information. If the authenticated information points to the public key, an identification/label should be stored with the public key for easy reference.
- *Long term data encrypting keys* must remain associated with the encrypted information. The data encrypting key should be stored with the encrypted information, or a pointer to the data encrypting key should be stored with the encrypted information. If the encrypted information points to the data encrypting key, an identification/label should be stored with the key for easy reference.
- Encrypted keys must remain associated with the key that will decrypt the encrypted key (e.g., a key encrypting key used for key wrapping, or a private key transport key). The two keys should either be stored together, or a pointer to the key encrypting key should be stored with the encrypted key. If the encrypted key points to the key encrypting key, an identification/label should be stored with the key encrypting key for easy reference.
- *Master keys used to derive other keys* may need to be available for the lifetime of any keys derived from the master key. The primary method for continuing availability is to make one or more copies of the master key and store each copy separately. A method should be provided for the association of the derived keys with the master keys (e.g., an identification/label).
- *Keys derived from a master key* may need to remain associated with that master key. The master key should be stored with the derived keys, or a pointer to the master key should be stored with the derived keys. If the derived keys point to the master keys, an identification/label should be stored with the master keys for easy reference.
- A *key transport private key* must be associated with the keying material that is transported using that key. Normally, the transported keying material will require a pointer to the private key with an appropriate identification or label.
- A *static key agreement private key* must be associated with its domain parameters to allow the calculation of shared secrets during the key agreement process. The domain parameters or a pointer to the domain parameters should be stored with the private key. If

the private key points to the domain parameters, an identification/label should be stored with the domain parameters for easy reference.

- *A static key agreement public key* must remain associated with its domain parameters to allow the calculation of shared secrets during the key agreement process. The domain parameters or a pointer to the domain parameters should be stored with the public key. If the public key points to the domain parameters, an identification/label should be stored with the domain parameters for easy reference.
- Public/private key pairs that were generated using *domain parameters* must remain associated with those domain parameters. The domain parameters or a pointer to the domain parameters should be stored with the keys. If the keys point to the domain parameters, an identification/label should be stored with the domain parameters for easy reference.
- An *initialization vector* must remain available to decrypt information that was encrypted using that IV, or to verify the integrity of a MAC that was computed using the IV. The IV or a pointer to the IV should be stored with the protected information. If the protected information points to the IV, an identification/label should be stored with the IV for easy reference.
- A *shared secret* may (or may not) need to remain associated with the keying material that is derived from the shared secret. If the association is required, the keying material should either be stored with the shared secret, or a pointer to the shared secret should be stored with the keying material. If the keying material points to the shared secret, an identification/label should be stored with the shared secret for easy reference.
- A *seed* may need to be associated with the information that was generated from that seed (e.g., the domain parameters). If the association is required, the seed should either be stored with the generated information, or a pointer to the seed should be stored with the information. If the generated information points to the seed, an identification/label should be stored with the seed for easy reference.
- An *intermediate result* may need to be associated with the process that uses that result until such time as the intermediate result is no longer required. Intermediate results should be securely stored with the process if storage is required.

4.7.2 Operational Storage

Keying material may need to be stored for normal cryptographic operations during the cryptoperiod of the key. The storage requirements of Section 4.7.1 apply to this keying material.

4.7.3 Backup Storage

The backup of keying material on an independent, secure media provides a source for key recovery (see Section 4.9). Backup refers to storage during operational use. Not all keys should be backed up. The storage requirements of Section 4.7.1 apply to keying material that is backed up. Table 2 provides guidance about the backup of each type of keying material; however, the final determination for backup should be made based on the application in which the keying material is used.

Table 2: Backup of Keying Material by Material Type

Type of Key	Backup?
Signing keys	No; non-repudiation would be in question.[However, it may be warranted in some cases - a CA's signing key, for example]
Signature verification keys	OK; its presence in a public-key certificate that is available elsewhere may be sufficient.
Secret authentication keys	OK
Private authentication key	OK, if required by an application.
Public authentication key	OK; its presence in a public-key certificate that is available elsewhere may be sufficient.
Long term data encryption keys	OK
Short term data encryption keys	May not be necessary
RNG keys	Not necessary and may not be desirable, depending on the application.
Key encrypting key used for key wrapping	OK
Master key used for key derivation	OK, unless a new master key can easily be generated and distributed.
Keys derived from a Master Key	Depends on the use of the derived key, but backup may not be needed if the master key is backed up.
Key transport private keys	OK
Key transport public keys	OK; presence in a public-key certificate available elsewhere may be sufficient.
Static key agreement private keys	No, unless needed for reconstruction during key recovery?
Static key agreement public keys	OK; its presence in a public-key certificate that is available elsewhere may be sufficient.
Ephemeral key agreement private keys	No
Ephemeral key agreement public keys	No, unless needed for reconstruction during key recovery?
Secret authorization key	OK
Private authorization key	OK
Public authorization key	OK; its presence in a public-key certificate that is available elsewhere may be sufficient.
Domain parameters	OK

Initialization vectors	OK, if necessary
Shared secrets	No, unless needed for reconstruction during key recovery?
Seeds	No, unless required for the validation of domain parameters
Intermediate results	No

4.7.4 Key Archive Storage

A key management archive is a repository containing keying material of historical interest. Not all keying material needs to be archived. Archived keying material may be either static (i.e., never changing) or may need to be re-encrypted under a new archive encryption key. Archived data should be stored separately from active data. Multiple copies of the archived data should be available and stored separately from each other. When no longer required, the keying material should be destroyed in accordance with Section 4.10.

Archived keying material requires confidentiality and/or integrity protection. Confidentiality is provided by an archive encryption key (one or more encryption keys that are used exclusively for the encryption of archived information) or by another key that has been archived. Likewise, integrity protection is provided by an archive integrity key (one or more authentication or digital signature keys that are used exclusively for the archive) or by another key that has been archived. The archive keys may be either symmetric (secret) keys, or private/public key pairs. The keys used for confidentiality and integrity should be different, and should be protected in the same manner as their key type (see Sections 3.3.1 and 4.7). [Note: May need to discuss random access protection vs. database protection?]

- *Signing key*: A private key used for signing should not be archived.
- *Signature verification key*: A public key used for the verification of signed information should be archived and protected in accordance with Sections 3.3.1 and 4.7 as long as any information signed with the associated private key is maintained. The key should be archived prior to the end of the validity period (i.e., cryptoperiod, reliance period) of the key. This public key should never be altered while in the archive until the key is destroyed.
- *Secret authentication key*: This secret key should be archived and protected in accordance with Sections 3.3.1 and 4.7 as long as any information protected by the key may need to be authenticated. The key should be archived when the protected data is archived, presumably when the data is not used actively. If the key is archived in an encrypted form, the key encrypting key (i.e., the archive key) should be an active key (i.e., the cryptoperiod of the archive encryption key should not have expired). At the end of the cryptoperiod of the archive encryption key, a new archive encryption key should be generated, and the authentication key should be re-encrypted using the new archive encryption key.
- *Private authentication key*: This key should not be archived, since it will no longer be needed.

- *Public authentication key*: This public key should be archived and protected in accordance with Sections 3.3.1 and 4.7 as long as any information protected by the key may need to be authenticated. The key should be archived when the protected information is archived, presumably when the information is not used actively. This public key should never be altered while in the archive until the key is destroyed.
- *Long term data encryption key*: This secret key should be archived and protected in accordance with Sections 3.3.1 and 4.7 as long as any information protected by the key may need to be decrypted. The key should be archived when the encrypted information is archived, presumably when the information is not used actively. If the key is archived in an encrypted form, the key encrypting key (i.e., the archive encryption key) should be an active key (i.e., the cryptoperiod of the archive encryption key should not have expired). At the end of the cryptoperiod of the archive encryption key, a new archive encryption key should be generated, and the long term data encryption key should be re-encrypted using the new archive encryption key.
- *Short term data encryption key*: There is no valid reason to archive this key.
- *RNG key*: This key should not be archived.
- *Key encrypting key used for key wrapping*: This secret key encrypting key should be archived and protected in accordance with Sections 3.3.1 and 4.7 as long as any keying material protected by the key encrypting key or any keying material protected by the encrypted keying material may need to be decrypted or authenticated. The key encrypting key should be archived by the end of the cryptoperiod of the key, providing that the encrypted keying material needs to be archived. If the key encrypting key is archived in an encrypted form, the archive encryption key should be an active key (i.e., the cryptoperiod of the archive encryption key should not have expired). At the end of the cryptoperiod of the archive encryption key, a new archive encryption key should be generated, and the key encrypting key should be re-encrypted using the new archive encryption key.
- *Master key used for key derivation*: A master key may need to be archived if there is a requirement to access that key in the future (e.g., to recreate the derived keys?). If the master key is archived, then it should be archived when no longer needed to actively derive keys; if stored in an encrypted form, the archive encryption key should be an active key (i.e., the cryptoperiod of the archive encryption key should not have expired). At the end of the cryptoperiod of the archive encryption key, a new archive encryption key should be generated, and the master key should be re-encrypted using the new archive encryption key.
- *Key derived from a master key*: A derived key may need to be archived, depending on its use. See the other key classes.
- *Key transport private key*: This key may need to be archived if needed to decrypt an archived data encryption key that is used to encrypt archived data. It might be easier to encrypt the data encryption key using the archive encryption key. However, if the key

transport private key is archived, the private key should be encrypted using an active archive encryption key (i.e., the cryptoperiod of the archive encryption key should not have expired). At the end of the cryptoperiod of the archive encryption key, a new archive encryption key should be generated, and the key transport private key should be re-encrypted using the new archive encryption key.

- *Key transport public key*: There is no need to archive this key.
- *Static key agreement private key*: This private key should not be archived unless needed for key recovery by reconstruction of the agreed upon keying material. If the private key is archived, the private key should be archived prior to the end of the validity period of the private key and should be protected in accordance with Sections 3.3.1 and 4.7. If the key is archived in an encrypted form, the archive encryption key should be an active key (i.e., the cryptoperiod of the archive encryption key should not have expired). At the end of the cryptoperiod of the archive encryption key, a new archive encryption key should be generated, and the static key agreement private key should be re-encrypted using the new archive encryption key. The private key should be destroyed when no longer needed for key recovery.
- *Static key agreement public keys*: This public key need not be archived unless needed for key recovery by reconstruction of the agreed upon keying material. The public key should be archived prior to the end of the validity period of the public key and should be protected in accordance with Sections 3.3.1 and 4.7. The public key should never be altered while in the archive until its destruction destroyed.
- *Ephemeral key agreement private keys*: This private key should not be archived.
- *Ephemeral key agreement public key*: This public key need not be archived unless needed for key recovery by reconstruction of the agreed upon keying material. The public key should be archived prior to the end of the validity period of the public key and should be protected in accordance with Sections 3.3.1 and 4.7. The public key should never be altered while in the archive until its destruction destroyed.
- *Secret authorization key*: This key should not be archived.
- *Private authorization key*: This key should not be archived
- *Public authorization key*: This key should not be archived
- *Domain parameters*: Domain parameters used for digital signature algorithms should be archived with protections as defined in Sections 3.3.1 and 4.7 if the signed information and signature verification key are archived, or if the public key pairs used in a key agreement scheme are archived for key recovery. The domain parameters should be archived prior to the end of the validity period (i.e., cryptoperiod, reliance period) of the signature verification key. Domain parameters should never be altered while in the archive until the signature verification keys are destroyed.
- *Initialization vectors*: An IV should be archived if the information protected using the IV is archived. The IV should be archived when the protected information is archived, with

protections defined in Sections 3.3.1 and 4.7. If the IV is archived in an encrypted form, the archive encryption key should be an active key (i.e., the cryptoperiod of the archive encryption key should not have expired). At the end of the cryptoperiod of the archive encryption key, a new archive encryption key should be generated, and the IV should be re-encrypted using the new archive encryption key.

- *Shared secret*: A shared secret should not be archived unless needed to validate or reconstruct the derived keying material. If archival is required, the shared secret should be archived when no longer actively needed to derive keying material, with protections as defined in Sections 3.3.1 and 4.7. If the shared secret is archived in an encrypted form, the archive encryption key should be an active key (i.e., the cryptoperiod of the archive encryption key should not have expired). At the end of the cryptoperiod of the archive encryption key, a new archive encryption key should be generated, and the shared secret should be re-encrypted using the new archive encryption key.
- *Seeds*: A seed should not be archived unless needed to validate or reconstruct pseudorandom numbers. If archival is required, the seed should be archived immediately, or when used as part of elliptic curve domain parameters, when the domain parameters are archived, with protections as defined in Sections 3.3.1 and 4.7. If the seed is archived in an encrypted form, the archive encryption key should be an active key (i.e., the cryptoperiod of the archive encryption key should not have expired). At the end of the cryptoperiod of the archive encryption key, a new archive encryption key should be generated, and the seed should be re-encrypted using the new archive encryption key.
- *Intermediate results*: Intermediate results should never be archived.

4.8 Key Update

Prior to or at the time of the end of a key's cryptoperiod, the key needs to be replaced by a new key if a cryptographic capability is to continue. A key may be replaced by rekeying, whereby a different key is established that does not depend (mathematically) on the key being replaced. This may be accomplished using the key establishment methods discussed in Section 4.4. Alternatively, the key may be replaced by a key update method, whereby the current key or a master key is modified to create a new key (the new key is derived from the old key or the master key). [Note: Need more guidance, including discussions about limiting the number of updates before rekeying, and the evaluation of update procedures.]

4.9 Key Recovery

The process of retrieving the keying material from backup or archive storage is called key recovery. Key recovery is a broad term that may apply to several different key recovery techniques. Each technique will result in the recovery of a cryptographic key and other information associated with that key (i.e., the keying material). The information required to recover that key may be different for each application or each key recovery technique. The term "Key Recovery Information" (KRI) is used to refer to the aggregate of information needed to recover the key. The KRI includes the key to be recovered (perhaps in an encrypted form or divided into multiple components - see Section 4.7.1.1) and other cryptographic data (e.g., IVs), the time when the key was created, the identity of the owner of the key (the individual, application or organization who created the key or who own the data protected by that key) and any conditions that must be met by a requestor to be able to recover the keying material.

[Note: A list of general requirements for a KRS can be provided.]

4.10 Key De-registration and Destruction

When there are no further requirements for retaining keying material or its association with an entity, the key should be de-registered (i.e., all records of the keying material and its associations should be destroyed), and all copies of the private or secret key should be destroyed. Any media on which the keying material was stored should be erased in a manner that removes all traces of the keying material so that it cannot be recovered by either physical or electronic means. [Note: while it may be desirable to destroy all copies of a public key in many cases, it is not possible to guarantee that this is actually done. Retention of the public key is not a security problem.]

4.11 Key Revocation

It may be 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 form an organization, etc. This is accomplished by notifying all entities that may be using the revoked keying material that the material should no longer be used. The notification should include a complete identification of the keying material, the date and time of revocation and the reason for revocation (e.g., key compromised). Based on the revocation information provided, the other entities could make a determination of how they would treat information protected by the revoked keying material.

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

As another example, a secret key used to generate MACs could be revoked so that it would not be used to generate MACs on new information. However, the key could be retained so that archived documents could be verified.

5. General Key Management Guidance

5.1 Key Management Policy

U. S. Government agencies that use cryptography are responsible for defining the Key Management Policy (KMP) that governs the lifecycle for the cryptographic keys as specified in Section 4. A Key Management Practices Statement (KMPS) is then developed based on the KMP and the actual applications supported. The KMPS should specify how key management procedures, and techniques are used to enforce the KMP. For example, a key management policy statement might be that secret and private keys must be protected from unauthorized disclosure. The corresponding key management practices statement might state that secret and private keys must be either encrypted or physically protected.

[Note: It is not intended that the Key Management Policy or the Key Management Practices Statement would create additional documentation for an agency to develop. The intention is to provide guidance on what should be included in currently required documentation.]

5.1.1 Key Management Practices Statement

5.1.2 Key Usage

A cryptographic key should be used for only one purpose. For example, a given symmetric key may be used for the encryption of data OR the encryption of keys (key wrapping) OR the creation of a Message Authentication Code OR the generation of random numbers, but should not be used for more than one of these functions. A public/private key pair may be used for signing and verifying digital signatures OR establishing keys, but not both. [Note: Additional guidance may be required.]

5.1.3 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 may 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, and
4. may limit the amount of time available for cryptanalytic attacks to be useful.

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

Among the factors affecting the risk of exposure are:

- the strength of the cryptographic mechanisms (e.g., the algorithm, key length, mode of operation),
- the embodiment of the mechanisms (e.g., FIPS 140-2 Level 4 implementation, software implementation on a Microsoft Windows machine),

- the operating environment (e.g., secure limited access facility, open office environment, publicly accessible terminal),
- the volume of information flow or the number of transactions,
- the security function (e.g., data encryption, digital signature, key production or derivation, key protection),
- the rekeying method (e.g., keyboard entry, rekeying using a key loading device where humans have no direct access to key information, remote rekeying within a PKI),
- the number of nodes in a network that share a common key, and
- the threat to the information (e.g., who the information is protected from, and what are their perceived technical capabilities and financial resources to mount an attack).

In some cases, increased risk may suggest shorter cryptoperiods, while, in other cases, increased risk may suggest a need for longer cryptoperiods,. For example, some cryptographic algorithms may be more vulnerable to cryptanalysis if the adversary has access to large volumes of stereotyped data that is encrypted under the same key. Particularly where a secret key is shared among several nodes, it may be prudent to employ short cryptoperiods for such algorithms. On the other hand, where the rekeying method is particularly subject to human error or other frailty, more frequent rekeying might actually increase the risk of exposure. It may be more important to have trusted expert invocation or supervision of the process than frequent repetition of the process.

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

To facilitate discussions, the following discussions are provided for the different key types:

1. *Signing key*: The cryptoperiod of a signing key should, in general, be shorter than the cryptoperiod of the corresponding signature verification key. For further advice, see Appendix A. [Need to provide guidance on the specific length of the cryptoperiod - depends on the amount of information to be signed - maximum about 1-2 years or XXX uses?]
2. *Signature verification key*: The cryptoperiod of a signature verification key should, in general, be longer than the cryptoperiod of the corresponding signing key so that signed information could be verified at a later time than when it was signed. For further advice, see Appendix A. [Need to provide guidance on the specific length of the cryptoperiod]

3. *Secret authentication key*: The cryptoperiod of a secret authentication key depends on the sensitivity of the type of information it protects. For very sensitive information (e.g., information that one would not like to be compromised (unprotected) if the key used to protect other information were compromised), should have an authentication key that is unique to that protected information. Otherwise, suitable cryptoperiods may extend beyond a single use.
4. *Private authentication key*: A private authentication key would presumably be used multiple times. It's associated public key could be certified, for example, by a Certificate Authority or Attribute Authority. The cryptoperiod of the private authentication key and its associated public key would be the same, e.g., for the cryptoperiod of the certificate. An appropriate cryptoperiod for the key would be 1-2 years, depending on its use.
5. *Public authentication key*: The cryptoperiod would be the same as the associated private authentication key. See the discussion for the *private authentication key*.
6. *Long term data encryption key*: A long term data encryption key is used multiple times over an extended period of time. An encryption key that is used to encrypt large volumes of information over a short period of time (e.g., for a link encryption) should have a relatively short cryptoperiod (e.g., a day or a week). An encryption key used to encrypt less information could have a longer cryptoperiod.
7. *Short term data encryption key*: The cryptoperiod of a short term data encryption key is very short, e.g., a single message or a communication session.
8. *RNG key*: The cryptoperiod of a key used to create random numbers depends on the amount of its use, though it may be used for an extended period of time. Suitable cryptoperiods might be a month or a year or two.
9. *Key encrypting key used for key wrapping*: This key may be used multiple times over an extended period of time. A key of this type that is used to encrypt large numbers of keys over a short period of time should have a relatively short cryptoperiod (e.g., a day or a week). If a small number of keys are encrypted, the cryptoperiod of the key encrypting key could be longer.
10. *Master key used for key derivation*: A master key is used multiple times. Therefore, a suitable cryptoperiod depends on the considerations provided earlier in this section.
11. *Keys derived from a master key*: The cryptoperiod of a key derived from a master key is relatively short, e.g., a single use or a communication session or transaction.
12. *Key transport private key*: A key transport private key would presumably be used multiple times. The cryptoperiod of the key transport private key and its associated public key would be the same, e.g., for the cryptoperiod of the certificate. An appropriate cryptoperiod for the key would be 1-2 years.
13. *Key transport public key*: The cryptoperiod would be the same as the associated key transport private key. See the discussion for the *key transport private key*.
14. *Static key agreement private key*: A static key agreement private key would presumably be used multiple times. The cryptoperiod of the private key and its associated public key would be the same, e.g., for the cryptoperiod of the certificate. An appropriate cryptoperiod for the key would be 1-2 years.

15. *Static key agreement public key*: The cryptoperiod would be the same as the associated static key agreement private key. See the discussion for the *static key agreement private key*.
16. *Ephemeral key agreement private key*: The cryptoperiod of an ephemeral key agreement private key should be the duration of a single key agreement process.
17. *Ephemeral key agreement public key*: The cryptoperiod of an ephemeral key agreement public key should be the duration of a single key agreement process.
18. *Secret authorization key*: An 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. Suitable cryptoperiods should be less than two years.
19. *Private authorization key*: 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. The cryptoperiod of the private authorization key and its associated public key should be the same. Suitable cryptoperiods should be less than two years.
20. *Public authorization key*: The cryptoperiod of the public authorization key should be the same as the private authorization key: less than two years.

Other keying material does not have well established cryptoperiods, per se.

- Domain parameters remain in effect until changed.
- An IV is associated with the information that it helps to protect, and is needed until the information and its protection are no longer needed.
- Shared secrets should be destroyed when no longer needed to derive keying material.
- Seeds should be destroyed immediately after use unless needed for validation (e.g., as one of the elliptic curve domain parameters).
- Intermediate results should be destroyed immediately after use.

5.1.3 Domain Parameter Validation and Public Key Validation

Domain parameter validation and key validation techniques are important to most applications of public key cryptography.

The domain parameters used for DSA, ECDSA and the key agreement algorithms should be generated by a trusted party (e.g., a CA), OR generated by someone else and validated by a trusted party or the participating entities themselves. The validation procedures for DSA and discrete log key agreement algorithms are provided in ANSI X9.42[9]. The validation procedure for the elliptic curve key agreement algorithm is provided in ANSI X9.63.

Signature verification keys (public keys) must be validated to ensure that they are indeed associated with a private key (signing key) known by the purported owner of the public key; this is commonly known as “proof of possession”. This process is accomplished by the validating party (e.g., a CA) presenting a document or a known value (a challenge) to the public/private key

[9] The DSA domain parameters have the same properties and are created in the same manner as those for ANSI X9.42, so can be validated using the same process.

owner, the owner signing the document or known value using the private key, and the validating party (e.g., the CA) verifying the signature using the public key. If the validating party is a trusted party, then the trusted party could provide assurance to other parties of the association between the signing and verification keys (e.g., by signing a public key certificate).

The public keys used for DSA, ECDSA and the key agreement algorithms must also be validated. Public key validation is discussed in [FIPS XXX][10]. [Note: Need to address the validation of public key transport keys.]

5.1.4 Compromise of Keys and other Keying Material

Information protected by cryptographic mechanisms are secure only if the algorithms remain strong, and the keys have not been compromised. Key compromise occurs when the protective mechanisms for the key fail (e.g., the confidentiality, integrity or association of the key to its owner fail - see Sections 3.3.1 and 4.7), and the key can no longer be trusted to provide the required security. When a key is compromised, all use of the key should cease, the compromised key should be revoked, and all entities using or relying on that key should be notified (see Section 4.11). 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 meanings:

- A compromise of the confidentiality of a key means that another entity (an unauthorized entity) may know the key and be able to use that key to perform computations requiring the use of the key.

  For example, if a secret key (symmetric 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 parties. The entity might masquerade as a legitimate entity and send false information, i.e., the authenticity and source of the information would be in question.

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

- A compromise of the integrity of a key means 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).

- A compromise of a key's usage or application association means that the key would be used for the wrong purpose (e.g., key establishment instead of signatures) or for the wrong application.

[10] FIPS XXX refers to ANSI X9.42 and X9.63 for public key validation; the DSA public keys are validated in the same manner as the public keys for ANSI X9.42.

- A compromise of a key's association with the owner or other entity means that the identity of the other entity cannot be assured (i.e., one doesn't know who the other entity really is) or that information cannot be processed correctly (e.g., encrypted or decrypted with the correct key).
- 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 "stuff".

In general, the compromise of a key used to provide confidentiality protection[11] (i.e., via encryption) means that all information encrypted by that key may be known by unauthorized parties. In addition, the encrypted information may contain false information that was originated by an unauthorized party (a party that is not authorized to know the key).

The compromise of a key used to provide integrity[12] calls into question the integrity of all information protected by the key. This information may have been provided by, or changed by, an unauthorized entity.

The compromise of an entity authentication key means that unauthenticated entities can be, or may have been falsely authenticated.

The compromise of an authorization key means that unauthorized entities may be given authorizations to which they are not entitled.

[Note: Discussion could be included on the effects of a compromise on each type of keying material.]

5.1.5 Accountability

The lifecycle of a cryptographic key may be quite complicated. From its initial generation, through its use and final destruction, a key may be available in both plaintext and ciphertext forms; it may reside in several different systems; it may be under the control of different individuals; and it may have been used to generate or protect many other keys. With any key management system, there is always the possibility that some of its keys may be compromised at some time. Therefore, it is important to consider how to mitigate the effects of a key compromise when it does occur.

Perhaps the worse form of key compromise is one that is not detected. Nevertheless, even in this case certain protective measures can be taken. Key management systems should be designed to mitigate the negative effects of a key compromise. One should avoid designing large systems where the security of all the user keys is compromised by the compromise of a single key. Many cryptographic keys only protect the data of a single user or a limited number of users. Often systems have alternative methods to authenticate communicating parties that do not rely solely on the possession of keys. The object is to avoid building a system with catastrophic weaknesses.

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

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

Accountability can be an effective tool to help prevent key compromises and to reduce the impact of compromises once they are detected. Accountability involves the identification of those who have control of, and access to, cryptographic keys throughout their lifecycles.

Accountability provides three significant advantages:

- It aids in the detection of where in the lifecycle the compromise may have occurred and what individuals may have been involved.
- It tends to protect against compromise because individuals with access to the key know that their participation in the key lifecycle is known.
- It is very useful in recovering from a detected compromise of a key to know where it was used and what data, or other keys, it protected.

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

- Uniquely identifying keys.
- Identifying key users.
- Identifying dates and times of key use along with the data that is protected.
- Identifying other keys that are protected by a secret or private key.
- Limiting the amount of time a secret or private key is in plaintext form.
- Restricting plaintext secret and private keys to approved key containers. This includes key generators, key transport devices, key loaders, cryptographic modules, and key storage devices.
- Preventing humans from viewing plaintext secret and private keys.
- Destroying keys as soon as they are no longer needed.

5.1.6 Audit

Key management systems should be periodically audited to ensure that the practices continue to effectively support the Key Management Policy. The protective mechanisms should be reassessed as to the level of security that they provide and are expected to provide in the future. New technology developments and attacks should be taken into consideration. Most importantly, the actions of the humans that use, operate and maintain the system should be reviewed to verify that they continue to follow established security procedures. Strong cryptographic systems can be compromised by lax and inappropriate human actions.

5.1.7 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 may be used to provide confidentiality, assurance of integrity, non-repudiation or access control. Policies should be established to address the protection and continued accessibility of cryptographically protected information, and procedures should be in place to ensure that the information remains viable during its lifetime.

For some applications, the keying material used to secure the information may need to be backed up (in backup storage - see Section 4.7.3) or archived (in archive storage - see Section 4.7.4) in order to either remove the protections on the information (e.g., decrypt the information) or verify that the information is as expected (e.g., verify that the integrity of the information has not be violated or that it is from the expected source). The process of retrieving the keying material from backup or archive storage is called key recovery.

Applications that may require these procedures include encrypted or authenticated information that is stored, or other applications where acquiring and distributing new keying material quickly would be cumbersome. Other applications may not need to save their keying material for an extended time because there are other procedures to recover their operational capability when the keying material or the information protected by the keying material becomes inaccessible. Applications of this type may include telecommunications where the transmitted information could be resent, or applications that could quickly acquire and distribute new keying material.

The primary purpose of backing up or archiving keying material (e.g., keys and IVs) is to be able to recover that material when it is not otherwise available to decrypt or check the information protected by the keying material. For example, encrypted information cannot be easily transformed into plaintext information if the decryption key is lost or modified; the integrity of data cannot be determined if the key used to verify the integrity of that data is not available. Recovery of the keying material is commonly known as key recovery, although cryptographic information other than keys may also need to be recovered. Key recovery is motivated by a need to recover or ascertain the validity of cryptographically protected information (e.g., the information that has been encrypted or authenticated) on behalf of an organization or individual.

5.1.7.1 Considerations for Key Recovery

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

[Note: Examples could be included advising whether or not key recovery might or might not be appropriate in certain scenarios and who may request a key recovery.]

5.1.7.2 Key Recovery Policy

An organization using cryptographic techniques to protect their information should develop a policy that addresses the protection and continued accessibility of that information. For each application and each cryptographic technique used, consideration should be given as to whether or not the keying material may need to be saved for later recovery of the keying material (e.g., the keys) to allow subsequent decryption or checking of the information protected by the keying material.

When it is determined that key recovery of keying material is required, a Key Recovery System (KRS) needs to be defined. The KRS should support the key recovery policy and consist of the techniques and facilities for saving and recovering the keying material, the policy for administering the system, and the personnel associated with the system. 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 backed up or archived information when required.

The policy should address (at a minimum):

- The keying material that needs to be saved for a given application. For example, keys and IVs used for the decryption of stored information protected by the keys and IVs may need to be saved. Keys for the authentication of stored or transmitted information may also need to be saved.
- How and where the keying material would be saved. For example, the keying material could be stored in a safe by the individual who initiates the protection of the information (e.g., the encrypted information), or the keying material could be saved automatically when the protected information is transmitted, received or stored. The keying material could be saved locally or at some remote site.
- Who will be responsible for protecting the backed up or archived keying material. Each individual, organization or sub-organization could be responsible for their own keying material, or an external organization could perform this function.
- Who can request key recovery and under what conditions. For example, the individual who protected the information or the organization to which the individual is assigned could recover the keying material. Legal requirements may need to be considered.
- How a request is authenticated and authorized.
- Under what conditions could the policy be modified and by whom.
- What audit capabilities and procedures would be included in the KRS. The policy should identify the events to be audited. Auditable events might include key recovery information requests and their associated responses; the startup and shutdown of audit functions; operations performed to read, modify or destroy the audit data; requests to access user authentication data; and the uses of authentication mechanisms.
- How the KRS would deal with aged keying material or the destruction of the keying material.
- Who would be notified when keying material is recovered and under what conditions. For example, the individual who encrypted data could be notified when the organization recovers the decryption key because the person is absent, but the individual might not be notified when the organization is monitoring the activities of that individual.
- What procedures need to be followed when the KRS or some portion of the data within the KRS is compromised.

5.2 Guidance for Cryptographic Algorithm and Key Size Selection

Cryptographic algorithms that provide the security services identified in Section 1.4 are specified in Federal Information Processing Standards (FIPS). Several of these FIPS-approved algorithms are defined for a number of key sizes.

5.2.1 Equivalent Algorithm Strengths

Cryptographic algorithms provide different “strengths” of security, depending on the algorithm and the key size used. In this document, two algorithms are considered to be of equivalent strength for the given key sizes if the amount of time needed to “break the algorithms” or determine the keys (with the given key sizes) is the same. The strength of an algorithm for a given key size is traditionally described in terms of the amount of time it takes to try all keys for a symmetric algorithm that has no short cut attacks (i.e., exhaust the key space). In this document, this equivalence is phased as providing “X bits of security”. [Note: Other metrics for algorithm equivalence exist.]

The user should be aware that the recommended key size equivalencies are based on assessments made as of the publication of this document. Periodic reviews will be performed to determine whether the stated equivalencies need revision.

Table 3 provides equivalence guidelines for the Approved algorithms.

- Column 1 indicates the number of bits of security provided by the algorithms and key sizes in a particular row.
- Columns 2 provides the symmetric key algorithms that provide the indicated level of security, where TDES is approved in [FIPS46-3] and specified in ANSI X9.52, and AES is specified in [FIPS197]. Note: it is assumed in the table that TDES is using three distinct keys.
- Column 3 provides the equivalent hash algorithms that are specified in [FIPS180-2] for the given level of security.
- Column 4 indicates the size of the parameters associated with the standards that use discrete logs (DSA as defined in [FIPS186-3] for digital signatures, and Diffie-Hellman (DH) and MQV key agreement as defined in ANSI X9.42 and [FIPS XXX]), where L is the size of the modulus p, and N is the size of q. The value of L is considered to be the key size.
- Column 5 defines the value for k (the size of the modulus n) for the RSA algorithm specified in [ANSIX9.31] and [PKCS1] and adopted in [FIPS186-3] for digital signatures, and specified in [ANSIX9.44] and adopted in [FIPSXXX] for key establishment. The value of k is commonly considered to be the key size.
- Column 6 defines the value of f (the size of n, where n is the order of the base point G) for the elliptic curve algorithms specified for digital signatures in [ANSIX9.62] and adopted in [FIPS186-3], and for key establishment as specified in [ANSIX9.63] and adopted in [FIPSXXX]. The value of f is commonly considered to be the key size.

Table 3: Equivalent strengths.

Bits of security	**Symmetric key algs.**	**Hash algs.**	**Discrete Logs (DSA, DH, MQV)**	**RSA**	**Elliptic Curves**
80		SHA-1	$L = 1024$ $N = 160$	$k = 1024$	$f = 160$
112	TDES		$L = 2048$ $N = 224$	$k = 2048$	$f = 224$
128	AES-128	SHA-256	$L = 3072$ $N = 256$	$k = 3072$	$f = 256$
192	AES-192	SHA-384	$L = 7680$ $N = 384$	$k = 7680$	$f = 384$
256	AES-256	SHA-512	$L = 15360$ $N = 512$	$k = 15360$	$f = 512$

5.2.2 Defining Appropriate Algorithm Suites

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

In many cases, a variety of key sizes may be available. For 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 recommendations that may be used to select an appropriate suite of algorithms and key sizes for Federal Government unclassified applications. A minimum of eighty bits of security are considered adequate for most applications until 2015. Thereafter, it is recommended that at least 112 bits of security be used.

- Column 1 indicates the years during which the algorithms specified in subsequent columns are appropriate for use.
- Column 2 identifies appropriate symmetric key algorithms and key sizes: the Triple DES algorithm (TDES) is specified in [FIPS46-3], the AES algorithm is specified in [FIPS197], and the computation of Message Authentication Codes (MACs) using block ciphers is specified in [MODES].
- Column 3 specifies the hash sizes to be used for most hash applications (e.g., digital signatures). Hash algorithms are specified in [FIPS180-2].

- Column 4 specifies the hash algorithm and minimum key size to be used for keyed-hash (HMAC) computations. HMAC is specified in [FIPS198].
- Column 5 indicates the minimum size of the parameters associated with DSA as defined in [FIPS186-3].
- Column 6 defines the minimum size of the modulus for the RSA algorithm specified in [ANSIX9.31]and [PKCS1] and adopted in [FIPS186-3] for digital signatures, and specified in [ANSIX9.44] and adopted in [FIPSXXX] for key establishment.
- Column 6 defines the minimum size of the base point for the elliptic curve algorithms specified for digital signatures in [ANSIX9.62] and adopted in [FIPS186-3], and for key establishment as specified in [ANSIX9.63] and adopted in [FIPSXXX].

Table 4: Recommended algorithms and minimum key sizes.

Years	**Symmetric key algs. (Encryption & MAC)**	**Hash Alg.**	**HMAC**	**DSA**	**RSA**	**Elliptic Curves**
Present - 2015	TDES AES-128 AES-192 AES-256	SHA-1 SHA-256 SHA-384 SHA-512	SHA-1 (≥80 bit key) SHA-256 (≥128 bit key) SHA-384 (≥192 bit key) SHA-512 (≥256 bit key)	Min.: L = 1024; N =160	Min.: k=1024	Min.: f=160
2016 and beyond	TDES AES-128 AES-192 AES-256	SHA-256 SHA-384 SHA-512	SHA-256 (≥128 bit key) SHA-384 (≥192 bit key) SHA-512 (≥256 bit key)	Min.: L = 2048 N = 224	Min.: k=2048	Min.: f=224

The algorithms and key sizes in the table should be considered appropriate for the protection of data during the given time periods. Algorithms or key sizes not listed for “2016 and beyond” should not be used to protect data during that time period. For example, if data is initially protected in 2014 and must remain secure for ten years (i.e., from 2014 to 2023), SHA-1 or a 1024 bit RSA key would not provide sufficient protection between 2016 and 2023. As another example, if a CA signature key and all certificates issued under that key will expire in five years, then the signature and hash algorithm used to sign the certificate needs to be secure for at least five years. A certificate issued in 2005 using 1024 bit DSA and SHA-1 would be acceptable; issuing a certificate in 2015 using the same algorithms would not provide adequate security for the certificate.

Algorithms of different strengths and key sizes may be used together for performance, availability or interoperability reasons, provided that sufficient protection is provided.

[Note: A list of algorithm combinations could be included that would discuss the security implications of the combination.]

Typically, an organization might select the cryptographic services that are needed for a particular application. Then, based on the security life of the data and the years the system is anticipated to be in use, it would select an algorithm and key size suite that is sufficient to meet these requirements. The organization would then establish a key management system including approved cryptographic products that provide the services required by the application. Finally, when the current suite nears its expiration date, the organization would transition to a new suite offering better security.

[Note: Examples could be given regarding appropriate algorithm and key size selections, given a desired level of security.]

5.2.3 Transitioning to New Algorithms and Key Sizes

There are many legacy applications currently available that use algorithms and key sizes not specified in Table 4. Information protected by these algorithms that must be retained should be updated or suitably archived to ensure the continued protection of that information. Likewise, information protected by an algorithm or key size that is considered to be adequate until 2015, but not beyond, should be updated or archived if protection is to be afforded after 2015.

It may be impossible to protect information that has been encrypted once the security life of the encryption algorithm, or its key, has expired (e.g., information that was encrypted using legacy applications). One must assume that encrypted data could have been collected and retained by unauthorized parties. At some time in the future when the cryptographic system becomes vulnerable, the unauthorized parties may attempt to decrypt the information. Even though the system has been replaced (e.g., by a different algorithm or key size), the previously encrypted information is vulnerable. Thus, when using Table 4 to select the appropriate key size for an encryption algorithm, it is very important to take the expected security life of the data into consideration.

[Note: Need to provide guidance about how to transition, including how to extend the protection on data that has been protected using “no longer secure” methods.]

5.3 Key Establishment Schemes

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

[Note: discussions will be included on using the schemes in FIPS XXX.]

6. Key Management Guidance - Selected Infrastructures

6.1 Public Key Infrastructure

6.2 Kerberos

7. Key Management Guidance - Selected Protocols

7.1 S/MIME

7.2 TLS/SSL

The Transport Layer Security (TLS) and Secure Socket Layer (SSL) standards are the primary end-to-end security protocols used to protect information on the Internet and World Wide Web. Their use is now ubiquitous, and these protocols are broadly implemented by web servers and browsers. In addition, SSL and TLS have been widely adapted to other applications, such as protecting connections to e-mail servers, or with directory servers. TLS is an enhanced and standardized version of SSL; the protocols are similar, but not identical. The Internet Engineering Task Force governs the Transport layer Security (TLS) standard [RFC2246]. The information in this section generally applies to both TLS and SSL, except where otherwise noted. Only the term TLS will be used in the text, except when it is necessary to make a distinction.

TLS is an end-to-end client/server protocol that sits on top of the Internet Transport Control Protocol (TCP) and depends upon TCP to provide a reliable end-to-end data service, specifically data is not lost, corrupted or received out of order. TLS will detect any such problems (in fact the cryptographic integrity service of TLS is much stronger than the integrity checks or TCP) but TLS will not attempt error recovery; it will simply abort the session with an error indication if integrity checks fail.

After first initiating a TCP connection, a TLS client, typically a web browser, establishes a TLS session with a server, typically a web server. For the duration of the session, TLS communication between the client and server provides integrity protection and (optionally) confidentiality. In addition, TLS can provide strong cryptographic authentication of the server, or both the server and the client, and this authentication is cryptographically bound with the integrity protection so that the entire session is authenticated. [Note: although the TLS protocol provides for "anonymous" integrity and authentication services, without any authentication, these services are not recommended for Federal use because they are subject to "man-in-the-middle" attacks.]

A TLS session begins with a series of "handshake" messages. The first is a Client Hello message to the server that states the client's capabilities. The server then selects the "cipher suite" to be used, and a shared master key is established (the master key is used to generate the keys to provide the confidentiality and integrity). The cipher suite selection and server actions determine which services will be used (in addition to integrity, which is always provided). In each case, the initial handshake ends with a key confirmation that confirms that both the client and server have the same master key and ensures the integrity of the sequence of handshake protocol messages up to this point. In addition, for every cipher suite recommended for Federal use, the server is cryptographically authenticated to the client, and in some of the recommended cipher suites the client is also strongly authenticated to the server.

TLS provides security for the duration of a communications session. Data is encrypted while in transit and decrypted when it is received. The server, and optionally the client, may be cryptographically authenticated as a part of the TLS session protocol. This authentication is bound to the entire session data contents. However, TLS does not provide a cryptographic non-repudiation service to allow validation of the session data or authentication at a later time by a third party.

7.2.1 Version

In general, an SSL/TLS negotiation takes place between the client application (the client) and the server application (the server): the client states its capabilities, and the server selects the protocol parameters from the client's capabilities. Today, most web servers and browsers are capable of supporting TLS; however, many servers are still running the earlier SSL versions of the protocol. The relation of TLS and SSL is expressed in version numbers: the final version of SSL uses the version number 3.0, while TLS 1.0 uses the version number 3.1.

Federal users should normally select the TLS protocol rather than the SSL protocol. The only way a client can enforce this selection is to implement only the TLS protocol, since the server can choose any of the protocols supported by the client. Similarly, Federal web servers should be configured to select the TLS protocol rather than the SSL protocol. In some cases, it may be necessary to accept an SSL session for interoperability; however, no version of SSL earlier than 3.0 should be used.

7.2.2 Cipher Suite Selection

TLS offers a number of cipher suites that use algorithms that are not approved for Federal use. Table 5 lists the TLS cipher suites that use Approved algorithms that may be used to protect Federal information. Note that TLS_RSA_WITH_NULL_SHA provides an authentication and integrity service, but does not provide confidentiality. Anonymous cipher suites are not recommended for use by Federal agencies, and none are included in Table 5. [Note: The shaded suites use AES encryption and, in anticipation of approval of the AES, these suites are defined in the IETF draft "AES cipher suites for TLS" <http://www.ietf.org/html.charters/tls-charter.html>]

Table 5: TLS cipher suites that may be used to protect Federal information

Suite	Key X	Auth	Encrypt	Digest
TLS_RSA_WITH_3DES_EDE_CBC_SHA	RSA	RSA	3DES-EDE-CBC	SHA-1
TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA	DH	DSS	3DES-EDE-CBC	SHA-1
TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA	DH	RSA	3DES-EDE-CBC	SHA-1
TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA	DH	DSS	3DES-EDE-CBC	SHA-1
TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA	DHE	RSA	3DES-EDE-CBC	SHA-1
TLS_RSA_WITH_NULL_SHA	RSA	RSA	NULL	SHA-1
TLS_RSA_WITH_AES_128_CBC_SHA	RSA	RSA	AES-128-CBC	SHA-1
TLS_DH_DSS_WITH_AES_128_CBC_SHA	DH_DSS	DSS	AES-128-CBC	SHA-1

TLS_DH_RSA_WITH_AES_128_CBC_SHA	DH_RSA	RSA	AES-128-CBC	SHA-1
TLS_DHE_DSS_WITH_AES_128_CBC_SHA	DHE_DSS	DSS	AES-128-CBC	SHA-1
TLS_DHE_RSA_WITH_AES_128_CBC_SHA	DHE_RSA	RSA	AES-128-CBC	SHA-1
TLS_RSA_WITH_AES_256_CBC_SHA	RSA	RSA	AES_256_CBC	SHA-1
TLS_DH_DSS_WITH_AES_256_CBC_SHA	DH_DSS	DSS	AES_256_CBC	SHA-1
TLS_DH_RSA_WITH_AES_256_CBC_SHA	DH_RSA	RSA	AES_256_CBC	SHA-1
TLS_DHE_DSS_WITH_AES_256_CBC_SHA	DHE_DSS	DSS	AES_256_CBC	SHA-1
TLS_DHE_RSA_WITH_AES_256_CBC_SHA	DHE_RSA	RSA	AES_256_CBC	SHA-1

Federal web servers should normally be configured to only use one or more of the cipher suites listed in Table 5. If security needs are moderate, and the broadest possible interoperability with the public is needed, Federal Web Servers may be configured to preferentially select one of the above suites if the suite is implemented in a client, or to select another suite offering 128-bit encryption when the client cannot implement any of the approved cipher suites.

Users configuring clients should bear in mind that the server makes the selection of the cipher suite from among the choices offered by the client. Therefore, if a client offers any weak cipher suite, the server may select it, even when the client indicates (by the order of its suites) that its preference is for another suite. Since less secure suites may offer greater performance, users may encounter servers that are configured to pick relatively insecure cipher suites over more secure alternatives. Moreover, while clients may allow users to configure the clients to disallow particular algorithms, the clients may not include any way for users to express their preferences among those suites that are allowed. Clients should not be configured to use any suites with 40- or 56-bit encryption.

Note that TLS is an on-line session protocol that does not create encrypted files that are saved; rather, data is encrypted at its source and decrypted at its destination. Therefore, key recovery is not normally required for TLS implementations. However, an eavesdropper can record a TLS session and attack it at a later time when more powerful technology is available. In choosing algorithm suites, Federal agencies should consider the value or sensitivity of the information being protected and how long it must be protected.

7.2.3 Public Key Certificates for TLS

Section 5.2.1 provides approximate equivalent public key strengths for 80, 112-, 128- and 256-bit symmetric key strengths, while Section 5.2.2 provides guidance on when 80-bit strength cryptography should no longer be relied upon. At a minimum, RSA and Diffie-Hellman subject public keys in certificates used with TLS for key transport or key agreement should be at least 1024 bits. This provides at least 80-bit equivalent strength. If protection is required after the end date specified in Section 5.2.2 for 80-bit strength, then a larger RSA or Diffie-Hellman subject public key is required. Similarly, ephemeral Diffie-Hellman keys used in TLS should also be at least 1024 bits, and, if protection is required after the date specified in Section 5.2.2, then keys larger than 1024-bits are required.

It is not necessary that the keys used to sign key-management certificates be as strong as the key-management keys themselves. It would be acceptable to issue and use a TLS server certificate with a 2005 expiration date and a 3072-bit RSA subject key, signed by a 1024-bit RSA key. However, clients should not accept any certificate signed with a 1024-bit RSA or DSA key after the date specified in Section 5.2.2. Key management certificate signature size is a PKI issue, and PKIs and root Certification Authorities should not issue certificates signed with 1024-bit keys and expiration dates after the dates specified in Section 5.2.2. Hashes used to sign TLS certificates should be consistent with signing key sizes, as given in Section 5.2.2.

Similarly, RSA or DSA subject public keys that are used in certificates only for authentication need only be large enough that they are secure at the time they are used in a TLS session, even if confidentiality protection is needed after the authentication keys can no longer be relied upon.

For example, suppose that a TLS session takes place in 2002, and confidentiality protection is required until 2020. The TLS_DHE_RSA_WITH_AES_128_CBC_SHA cipher suite might be selected. A 1024-bit RSA key would be sufficient for authentication in 2002, but a 1024-bit ephemeral Diffie-Hellman key agreement key would not be sufficient to provide confidentiality protection until 2020. But 2048-bit ephemeral Diffie-Hellman keys, and authentication certificates with 1024-bit RSA keys would provide adequate protection.

The combination of Diffie-Hellman ephemeral key agreement with RSA or DSA authentication provides two desirable security properties:

- Authentication certificates are used only for authentication, not for both authentication and key management.
- Since an ephemeral key is used only once, if any ephemeral Diffie-Hellman private key is compromised, only the one TLS session that used that key is also compromised. If a RSA or static Diffie-Hellman private key is compromised, every TLS session that used that key and certificate is compromised.

Therefore ephemeral Diffie-Hellman key agreement with either DSA or RSA authentication may be the most suitable cipher suites for security critical applications, although these options do not generally offer the best performance of the TLS cipher suites.

[Issues:

1. *Server identity:* Server certificates use the domain name of the server as the subject name. However, in many implementations, the SSL/TLS API mimics the BSD sockets API, and requires the IPaddress, not the domain name. Therefore, the SSL/TLS code itself can't check the name of the server. Do we need to require that the TLS implementation check the identity of the server?
2. *Use of the same key for authentication and key management:* In its most commonly used modes of operation, the same RSA certificate and key is used for both server authentication and key transport. TLS also allows the use of a Diffie-Hellman certificate for client authentication; the ability to generate a common master_secret authenticates the client. As a general rule, it is desirable to use different keys for authentication and key management. Should these modes that use keys for both authentication and key agreement be discouraged or prohibited for Federal use?

3. *Payload HMAC issues for 112-, 12-8 and 256-bit strength cryptography*: SHA-1 is used to construct the HMAC used to authenticate the payload in all cipher suites in Table 5. Does the principle of comparable strength require the use of an HMAC based on SHA-256, SHA-256 and SHA-512 when 112-bit, 128-bit or 256-bit strength is required?
4. *Verify-data field for 112-, 128- and 256-bit strength*: Is the 96-bit verify data field that is used to authenticate the handshake and for key confirmation sufficient for comparable strength when 112-, 128- and 256-bit strengths are selected?
5. *The TLS Pseudorandom Function for 112-, 128- and 256-bit strength*: The TLS Key Derivation Function uses a pseudorandom function (PRF) to generate an arbitrary length string of pseudo-random bytes. The PRF expands a secret, a label (specified by the TLS standard) and a seed using an iterated HMAC. Both SHA-1 and MD5 HMACs are used, and their results are x-ored to get the final output. However, the secret is split into halves, and one half is processed with MD-5, while the other half is processed with SHA-1; thus, if MD5 were sufficiently weak, some of the contribution of half the secret would be lost in the final random output. Moreover, the "comparable" strengths of SHA-1 and MD-5 are 80- and 64-bits respectively. Are they adequate in this application for 112-, 128- or 256-bit strengths?

8. Key Management Guidance - Selected Applications

8.1 Encrypted File Storage

8.2 ???

APPENDIX A: Cryptoperiods for Signing Key Pairs

APPENDIX X: References

[FIPS46-3]	Data Encryption Standard (DES), October 25, 1999
[FIPS140-2]	Security Requirements for Cryptographic Modules, May 25, 2001
[FIPS180-2]	Secure Hash Standard (SHS), [Insert Date]
[FIPS186-2]	Digital Signature Standard (DSS), January, 2000.
[FIPS197]	Advanced Encryption Standard (AES), [Insert Date]
[FIPS198]	Keyed-Hash Message Authentication Code (HMAC), [Insert Date]
[FIPS XXX]	[Key Establishment Schemes Standard], [Insert date]
[MODES]	Special Publication 800-XXX, Recommendation for Block Cipher Modes of Operation, [Insert date]
[SP800-5]	A Guide to the Selection of Anti-Virus Tools and Techniques, December 1992.
[SP800-21]	Guideline for Implementing Cryptography in the Federal Government, November, 1999.
[ANSI-X9.31]	Digital Signatures Using reversible Public Key Cryptography for the Financial Services Industry (rDSA), 1998
[ANSI-X9.42]	Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using discrete Logarithm Cryptography, 2001
[ANSI-X9.44]	Public Key Cryptography for the Financial Services Industry: Key Agreement Using Factoring-Based Cryptography, [Insert Date]
[ANSI-X9.52]	Triple Data Encryption Algorithm Modes of Operation, July, 1998
[ANSI-X9.62]	Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1998
[ANSI-X9.63]	Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, [Insert Date]
[PKCS1]	PKCS #1 v2.0: RSA Cryptography Standard, RSA Laboratories, October, 1998.
[HAC]	Handbook of Applied Cryptography, Menezes, van Oorschot and Vanstone, CRC Press, 1996.
[AC]	Applied Cryptography, Schneier, John Wiley & Sons, 1996.
[IPKI]	Introduction to Public Key Cryptography and the Federal PKI Infrastructure; Kuhn, Hu, Chang and Polk; draft NIST Special Publication, 2001 [available at http://csrc.nist.gov/publications]